AUFBAU UND TEST EINES DATENAUFNAHMESYSTEMS

RALF HERZBERG

Experimentelle physikalische Diplomarbeit
am I. Institut für Experimentalphysik
des Fachbereichs Physik
der Universität Hamburg

2. Mai 1989

Inhaltsverzeichnis

Teil I

Teil II

Benuzter-Manual

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Datenaufnahmesysteme sind in der wissenschaftlichen Forschung nicht nur in der Physik sondern auch z. B. in der Medizin und in der Umweltforschung, ein wichtiger Bestandteil in experimentellen Anordnungen. Die Funktionen solcher Systeme sind die Ausgabewerte von Meßinstrumenten aufzunehmen und zu verarbeiten. Die konkrete Aufnahme aller Meßdaten erfolgt nach einem Triggersignal, das dem System zugeführt werden muß. Die Bedingungen, die zu einem Triggersignal führen, nennt man Ereignis oder *event*. Die wesentlichen Aufgaben eines Datenaufnahmesystems lassen sich folgendermaßen charakterisieren.

* Erstens ist es notwendig die aufgenommenen Daten *online*, d.h. während der Messung, zu kontrollieren, weil auf anderem Wege eine Überwachung des Experiments nicht möglich ist. Nur so können Fehler während der Messung entdeckt und sofort korrigiert werden.
* Zweitens müssen die Meßwerte unverändert gespeichert werden, damit es möglich ist *offline*, d.h. nach der eigentlichen Messung und Datenaufnahme, die Meßdaten in der selben Reihenfolge, wie sie während der Messung aufgenommen wurden, wieder abrufen zu können. Dies ist erforderlich, wenn die verschiedenen Meßdaten erst kombiniert ausgewertet ein Endresultat liefern und die optimale Kombination zum Zeitpunkt der Datenaufnahme noch nicht bekannt ist, was in allgemeinen der Fall ist.

Je nach den spezifischen Anforderungen, die aus dem jeweiligen Experiment erwachsen, werden die eben genannten Aufgaben unterschiedlich gewichtet, so daß die Datenaufnahmesysteme entsprechend der Gewichtung ein sehr unterschiedliches Leistungsangebot aufweisen. Ein universelles System, das alle möglichen Forderungen gleichermaßen erfüllen kann, gibt es nicht, und wird es vorraussichtlich in nächster Zukunft auch nicht geben.

Auch in den Experimenten der Kern- und Hochenergiephysik, in denen es primär um den Nachweis von Teilchen und ihrer Energie geht, sind die Anforderungen an ein Datenaufnahmesystem nicht einheitlich. Während überwiegend in der Kernphysik bei hoher *event*-Rate eine sehr gute Energieauflösung verlangt wird, wobei die Datenmenge pro Ereignis im allgemeinen gering ist, ist die Gewichtung in der Hochenergiephysik meistens umgekehrt. So werden z.B. am H1-Experiment bei HERA pro *event* tausende von Einzelinformationen anfallen, die dann verarbeitet werden müssen.
Ein Komponente des H1-Detektors ist das PLUG-Kalorimeter [TPR87], das im extremen Vorwärtswinkelbereich das Flüssig-Argon-Kalorimeter ergänzt und mit Siliziumdetektoren instrumentiert wird. Es wird eines der ersten in der Hochenergiephysik eingesetzten derartigen Kalorimeter sein. Dabei sind umfangreiche Testmessungen erforderlich, um unter anderem die Besonderheiten die bei der Verwendung von Siliziumdetektoren auftreten können, zu studieren. Bislang wurden derartige Messungen mit elektromagnetischen Kalorimeteranordnungen durchgeführt. [BOR87]. Diese müßten auf die Untersuchung hadronischer Schauer erweitert werden [SIC89]. Im Zusammenhang mit der Entwicklung dieses Moduls müssen im Vorwege kalorimetrische Messungen mit Oberflächensperrschicht-Detektoren durchgeführt werden. Ein dafür geeignetes Aufnahmesystem für elektromagnetische Schauer sollte etwa 50 Detektoren auswerten können. Folgende Forderungen sind hierfür zu erfüllen:

Monitorfunktion während des Experiments

Für jeden Detektor muß die deponierte Energie *online* dargestellt werden könne, um die longitudinale Schauerentwicklung während des Experiments beobachten zu können. Außerdem muß ebenfalls *online* ein Spektrum aus der Summe aller pro *event* deponierten Energien gebildet werden. Auf diese Weise kann bereits während des Experiments die mittlere Gesamtenergie sowie die mit dem Kalorimeter erzielte Energieauflösung beobachtet werden. Nur so ist es möglich Fehler rechtzeitig zu erkennen und zu korrigieren.

Offline-Auswertung im *stand-alone*-Betrieb

Eine Auswertung *offline* sollte mit dem System vor Ort möglich sein, damit die knappen Strahlzeiten effektiver genutzt werden können und damit eine Abhängigkeit von Großrechenanlagen, auf denen sonst eine Auswertung erfolgen müßte, vermieden werden kann.

Kompatibilität mit Großrechenanlagen

Dennoch sollten die Datenbänder auch auf Großrechenanlagen gelesen werden können, damit eine Auswertung der Daten nicht auf ein einziges System beschränkt wird.

Zusätzliche Forderungen

Das System sollte leicht bedienbar sein, so daß die Benutzung nicht nur auf "Insider" eingekreist ist.
Schließlich muß es transportabel sein, damit Messungen an verschiedenen Orten (am DESY oder am CERN) möglich sind.

Für die Erfüllung dieser speziellen Aufgaben ist ein Rückgriff auf im Institut vorhandene Datenaufnahmesysteme, wie z.B. das von Strauß und Funk [FUN89], das System Phalst von Puskeppel und Niecke [PUS83], oder das Verarbeitungssystem von Vogel [VOG85], nicht sinnvoll, weil sie für andere spezielle Anforderungen entworfen wurden. Eine Anpassung der vorhandenen Systeme ist entweder nicht möglich oder wäre aber zu aufwendig. Deshalb wurde ein neues System konzipiert, das optimal an der vorliegenden Problematik orientiert ist, und dessen Aufbau und Test Thema dieser Arbeit ist.
Die Entwicklung eines solchen Datenaufnahmesystem bis zu der hier realisierten Version erstreckte sich von März bis Dezember 1988. Innerhalb dieser Zeit wurde es zur Durchführung umfangreicher Messungen in Experimenten am Isochronzyklotron eingesetzt. Über den physikalischen Zweck hinaus, konnten diese Experimente als Systemtest gewertet werden. Die dabei gewonnene Erfahrung wurde in der weiteren Entwicklung umgesetzt. (Teil I, Kap. 4 und 7). Ein abschließender funktionaler Test erfolgte im Januar 1989 (Teil I, Kap. 5 und 6). Aus pragmatischen Gründen ist diese Arbeit in zwei Teile gebunden. Der erste Teil ist folgendermaßen gegliedert:
In den Kapiteln 2 und 3 wird das System und seine Arbeitsweise vorgestellt.
Im Kapitel 4 wird eine Erweiterung des Systems beschrieben, die im engeren Sinn nicht zu dem Aufgabenfeld der Datenaufnahme gehört. Sie war am Anfang nicht vorgesehen, sondern wurde erst im Laufe der Arbeit entwickelt.
Kapitel 5, 6 und 7 beschreiben verschiedene Tests, die mit dem System ausgeführt worden sind.
In einer abschließenden Zusammenfassung werden die wichtigsten Ergebnisse dieser Arbeit noch einmal dargestellt.
Der zweite Teil stellt ein Benutzerhandbuch dar.

Kapitel 2

Die Systemarchitektur und das Betriebssystem RT11-SJ

Das Datenaufnahme-System (Data Acquisition-, DAQ-System) wird durch
* den Prozessor und die Peripherie,
* das Betriebssystem und
* die Benutzerprogramme gebildet.

In diesem Kapitel werden die Punkte 1 und 2 behandelt. Im Kapitel 3 werden die Benutzerprogramme vorgestellt.

2.1 Der Prozessor und die Peripherie

Bei der Wahl des Prozessors und der Peripherie kamen folgende Gesichtspunkte zum Tragen:
Wie schon in der Einleitung erwähnt wurde, bildeten die Anforderungen, die aus dem Einsatzfeld resultieren, die primären Gesichtspunkte. Die wichtigsten Anforderungen lauteten:

* Bis zu ca. 50 ADC-Daten (Analog-Digital-Converter) sollen pro *event* für eine *offline-Auswertung* im *list-mode*[1] abgespeichert werden.
* Bis zu ca. 50 ADC-Daten sollen pro *event* energiegeeicht in Monitorspektren und in einem Summenspektrum gespeichert werden.
* Ein hoher Durchsatz an Daten ist erstrebenswert aber nicht zwingend notwendig.

Außerdem sollte das DAQ-System
* kompakt und transportabel sein,
* aus Standardkomponenten bestehen, so daß in einem Schadensfall ein Ersatz kurzfristig beschafft werden kann, und schließlich sollte das System
* ein vertretbares Kosten-Nutzenverhältnis nicht überschreiten.

---

1) *list-mode* bedeutet sequentielles Speichern der ADC-Daten *event* für *event*.

Die unter diesen Gesichtspunkten resultierende System-Architektur wird in der Abbildung 2.1 schematisch dargestellt.

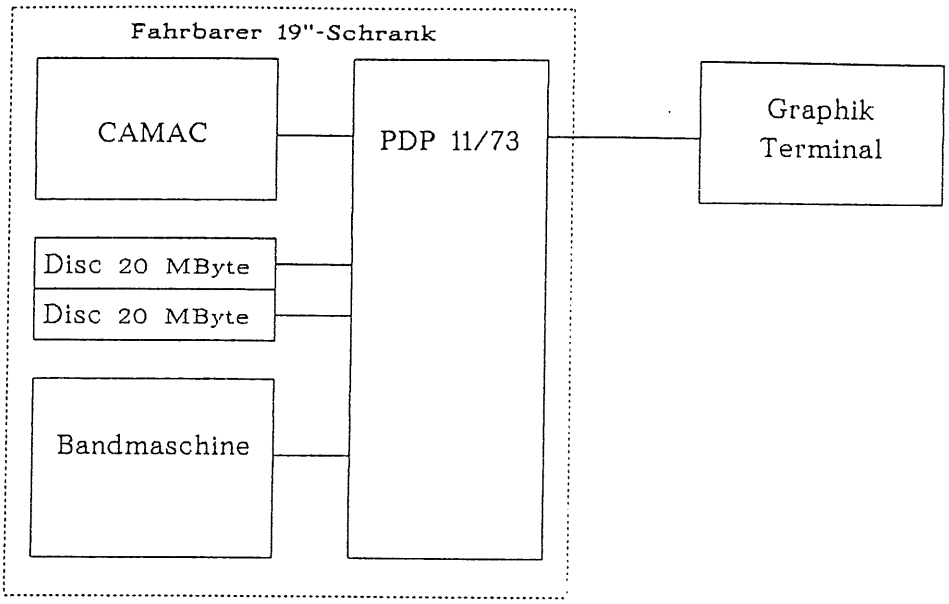


Abbildung 2.1: Die System-Architektur

Die PDP 11/73 ist ein DEC-Rechner mit dem KDJ11-A CPU Modul, auf dem die Benutzersoftware läuft. Mit diesem einzigen Prozessor wird die Datenaufnahme, -speicherung und -auswertung überwacht und durchgeführt.
CAMAC steht für "Computer Automated Measurement And Controll" und umfaßt Hardware- und Software-Spezifikationen für eine spezielle Schnittstellenerweiterung von Rechnersystemen. Die Hardware-Spezifikation sieht vor, daß in einem speziellen 19"-*Crate* bis zu 23 Instrumente wie z.B. ADCs eingesetzt werden können. Durch die Software-Spezifikation wird gewährleistet, daß Baugruppen gleichen Typs aber von verschiedenen Herstellern weitgehend kompatibel und austauschbar sind.
Auf den beiden Festplatten, die gleichberechtigt nebeneinander benutzt werden, sind u.a. Betriebs- und Benutzersoftware abgelegt.
Wegen der hohen Datenmenge werden die *list-mode*-Daten mit der Bandmaschine (Streamer-Tape-Unit) auf Magnetbänder geschrieben. Die Streamer-Tape-Unit kann besonders schnell schreiben und lesen.
Diese Architektur ist ähnlich der des Phalst-Systems von Puskeppel und Niecke [PUS83]. Anders jedoch als beim Phalst-System werden die Spektren nicht auf einem separaten Lifetime-Monitor dargestellt, sondern auf einem Graphikterminal gezeigt. Zu einem Lifetime-Monitor gehört eine zusätzliche Prozessorkarte, die mit dem Bus des Rechners verbunden wird und den Bildschirmaufbau organisiert. Für die Plazierung eines Punktes auf dem Monitor werden die entsprechenden Information über den Datenbus zu dieser Prozessorkarte gesendet. Diese Datenübertragung benötigt nur ca. 1 µs.

Beim Graphikterminal wird ebenfalls ein zusätzlicher Prozessor benötigt. Dieser ist jedoch außerhalb der PDP 11/73 in dem Terminal. Hier werden die Daten seriell übertragen. Eine Pixelplazierung benötigt dann 5156 µs[2]. Allerdings wäre eine schnellere Datenübertragung zu dem Graphikterminal nicht sinnvoll, weil es die Informationen nicht schneller verarbeiten könnte. Hierin liegt ein weiterer Unterschied zu einem Lifetime-Monitor. Während beim Lifetime-Monitor nur Pixelinformationen gesendet werden können, gibt es für das Graphikterminal ein komplette Kommandosprache mit der es möglich ist auch z.B. Ziffern und Buchstaben auf den Bildschirm zu bekommen. Dieser deutliche Nachteil wird jedoch durch 3 Punkte wieder kompensiert:

* Wenn während der Datenaufnahme auf die graphische Darstellung des Spektrums verzichtet wird, fällt die Übertragungszeit vollständig weg. Einzelheiten hierzu werden im Kapitel 3 beschrieben.
* Ein Graphikterminal ist wesentlich kostengünstiger.
* Das System wird kompakter, weil ein Ausgabegerät weniger benötigt wird, denn das Graphikterminal dient gleichzeitig als Bedienungsterminal.

2.2 Das Betriebssystem RT11-SJ

Unter den Betriebssystemen, die auf der PDP 11/73 verwendet werden können, wie z.B. RT11-XM, RT11-FB, oder RSX, ist das gewählte RT11-SJ dasjenige, unter dem die Benutzersoftware am schnellsten läuft. Ein Nachteil dieses Betriebssystems ist die eingeschränkte Zugriffsmöglichkeit auf das RAM (Random Access Memory). Dieser Kernspeicher hat bei der hier verwendeten PDP 11/73 eine physikalische Größe von 1 MByte. Unter RT11-SJ kann jedoch nur auf 64 KByte[3] zugegriffen werden, weil als Preis für die Rechengeschwindigkeit nur die unteren 64 KByte adressiert werden. Da in diesen 64 KByte ein Bereich für das Betriebssystem reserviert ist, stehen dem Benutzer insgesamt nur ca. 50 kByte zur Verfügung.

---

2) In dem Kapitel 7 wird diese Übertragungszeit berechnet.

3) Da in der Informatik häufig Zahlen verwendet werden, die eine Potenz von 2 darstellen, wurde folgende Konvention eingeführt:

1 KByte = 1024 Byte, aber 1 kByte = 1000 Byte,

wobei gilt 1 Byte = 8 Bits.

Um dennoch umfangreiche Programme realisieren zu können, muß eine besondere Programmtechnik, das Overlay, eingesetzt werden. Mit ihr wird der zur Verfügung stehende Speicherplatz **mehrfach** genutzt. Das Prinzip dieser Technik ist, nicht ständig das gesamte Benutzerprogramm im RAM geladen zu haben, sondern nur die Teile eines Programms zu laden, die tatsächlich zum momentanen Zeitpunkt in dem Programmfluß benötigt werden. Wenn dann zu einem späteren Zeitpunkt ein neues Programmteil geladen werden muß, wird dafür ein entbehrlicher Teil überschrieben. Mit dieser Technik kann z.B. das Programm DAQOUT (siehe auch Kap. 3) mit einer gesamten Maschinencodelänge von 92 kByte auf einen Speicherplatz von 49 kByte (53 %) untergebracht werden. Trotz dieser Technik war es dennoch nicht möglich, alle Aufgabenfelder des Datenaufnahmesystems innerhalb eines Programms zu vereinigen. Deshalb wurden die Aufgaben auf mehrere Programme verteilt. Es gibt 10 Programme, die zusammen die Datenaufnahme und -auswertung organisieren. Sie werden im Kapitel 3 behandelt. Für den Informations-Austausch zwischen den Programmen werden Dateien auf dem Plattenspeicher angelegt, so daß von jedem Programm aus auf diese Dateien zugegriffen werden kann.
Eine weitere Konsequenz der Speicherplatzeinschränkung ist die Begrenzung des für die Monitorspektren reservierten Raumes auf 30 KByte. Daher wurde das DAQ-System so ausgelegt, daß maximal 56 ADCs ausgelesen werden können.

Kapitel 3

Datenstruktur und Benutzerprogramme

Die Programme sind in den Programmiersprachen Fortran und Macro Assembler [KDJ84] geschrieben worden, wobei vor allen Dingen die zeitkritischen Programmteile in Macro Assembler geschrieben worden sind. Außerdem wurde bei der Programmierung dieser Teile darauf geachtet, daß während der Laufzeit nicht andere Programmteile geladen werden (siehe auch "Overlay", Kapitel 2.2), weil dadurch das Zeitverhalten stark beeinträchtigt werden würde.
Bevor die Programme im einzelnen erläutert werden, werden zwei wesentliche Mechanismen des DAQ-Systems beschrieben. Es handelt sich dabei um

* die Registrierung eines *events* und
* um die Speicherung der *list-mode*-Daten.

3.1 Die *event*-Registrierung

Für das Datenaufnahmesystem besteht ein *event* aus einer Mehrzahl von analogen Signalen, welche die zu messende Information beinhalten, und einem logischen Triggersignal, das dem DAQ-System mitteilt, daß ein *event* vorhanden ist. Die analogen Signale werden auf die *line-inputs* der ADCs gegeben, das Triggersignal auf den Eingang des LAM-Generators (Eigenentwicklung aus unserem Institut), der es unverzögert an die *gate-inputs* der ADCs weiterleitet (siehe auch Abbildung 3.1). Das Triggersignal besteht aus einem Rechtekkimpuls geeigneter Länge, und die analogen Signale müssen innerhalb dieses Zeitintervalls (Gate-Zeit) den ADCs zugeführt werden. Dies muß durch die vor den ADCs liegende Elektronik gewährleistet werden. Es gibt zwei Arten von ADCs, nähmlich *peak*-sensitive und *charge*-sensitive. Innerhalb der Gate-Zeit werden nun je nach ADC-Art entweder das Maximum des Signalstroms gesucht oder die Ladung durch Integration über die Zeit gebildet. Anschließend werden diese Werte in digitale Werte konvertiert. Bei den hier verwendeten ADCs liegen diese Zeiten zwischen 106 µs (Le Croy 2259B) und 60 µs (Le Croy 2249A). Zum Auslesen der ADCs generiert der LAM-Generator ein sogenanntes LAM-Signal (Look at me) und leitet es an den Crate-Controller weiter. Das LAM-Signal ist ein CAMAC-spezifisches Signal. Damit die ADCs nicht vor Ablauf der Konvertierungszeit ausgelesen werden, muß die Generierung des LAM-Signals verzögert werden. Die Verzögerung (1 ... 255 µs) ist über die Benutzersoftware programmierbar.

Nach einem Triggersignal am *gate-input* verriegeln sich die ADCs, so daß nachfolgende Triggersignale ignoriert werden. Erst das Löschen der Datenregister der ADCs sensibilisiert die ADCs erneut.

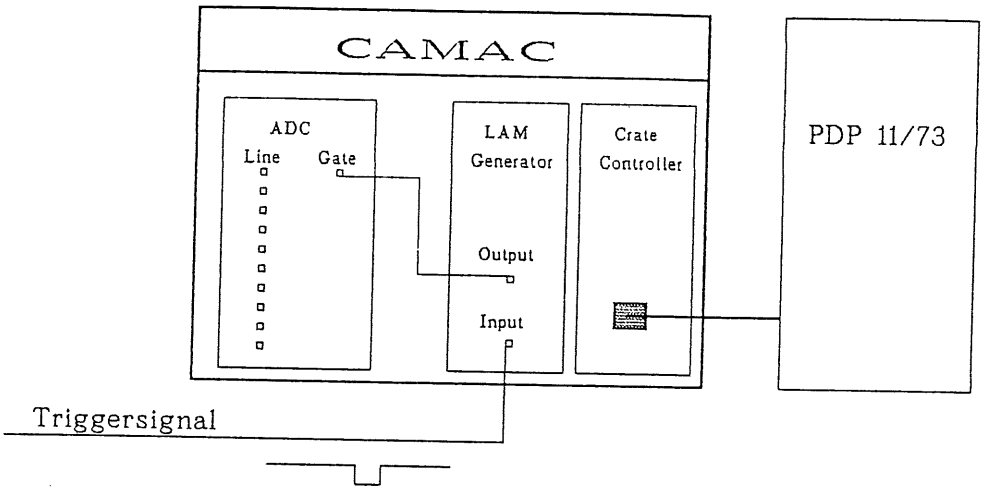


Abbildung 3.1: *Event*-Registrierung des DAQ-Systems

Wenn nun ein LAM-Signal von dem LAM-Generators generiert wird, so wird das LAM-Signal durch setzen eines Bits in dem Demand-Mask-Register (DMR) des Crate Controllers gespeichert. Die Datenaufnahmeprogramme können dieses Register kontrollieren und prüfen, ob jenes Bit gesetzt ist[1]. Wenn auf diese Weise ein *event* nachgewiesen wurde, dann springt das Programm in eine *event*-Verarbeitungsroutine, die hier CAMAC-Routine genannt wird. Das Flußdiagramm der CAMAC-Routine wird in der Abbildung 3.2 gezeigt.
Die "programmspezifischen Operationen" werden weiter unten (siehe Kap. 3.3) behandelt. Die übrigen Anweisungen sind CAMAC-Befehle und haben folgende Funktionen:

---

1) Prinzipiell bietet die CAMAC-Spezifikation eine weitere Möglichkeit an, ein LAM-Signal an ein Programm weiterzugeben, nähmlich indem das LAM-Signal ein *interrupt* auslöst. Dadurch würde der Programmablauf unterbrochen, und das Programm sprünge zur *event*-Verarbeitung in eine *interrupt*-Routine, nach deren Abfertigung das Programm an der unterbrochenen Stelle des alten Ablaufs fortführe. Da nun bei dem größten Teil der hier behandelten Software im Zusammenspiel mit der zu erwartenden *event*-Rate der Programmablauf durch *interrupts* erheblich gestört werden könnte, müßten für die betroffenen Programmteile ein *interrupt* unterdrückt werden. Das wäre programmierbar, trotzdem wäre der verbleibende Teil so klein, daß die Verwendung des *interrupt-mode* nicht sinnvoll wäre.

* "Disable LAM" bewirkt, daß der LAM-Generator weitere Triggersignale ignoriert und sie nicht an die ADCs weitergibt.
* "Clear LAM" setzt das Bit im Demand-Mask-Register zurück und löscht das LAM-Signal des LAM-Generators.
* "Clear Data" löscht die ADC-Datenregister und entriegelt die ADCs für das kommende Triggersignal.
* "Enable LAM" entriegelt den LAM-Generator.

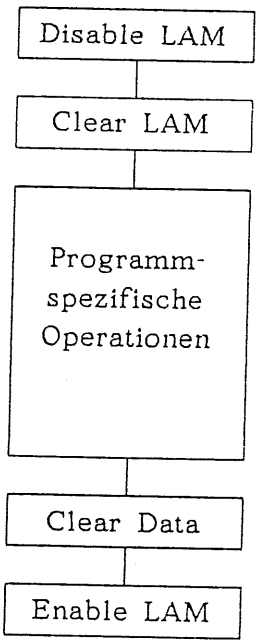


Abbildung 3.2: Flußdiagramm der CAMAC-Routine

Nachdem Durchlauf der CAMAC-Routine ist also ein *event* registriert, d.h. die ADCs haben die analogen Signale konvertiert, und die Daten sind an das Datenaufnahmeprogramm übermittelt worden.

3.2 Speicherung der *list-mode*-Daten

List-mode-Daten werden generell für die *Offline*-Auswertung in einem Massenspeicher festgehalten. Im allgemeinen werden hierfür Magnetbänder als Speichermedium verwendet, die zusammen quasi eine unbegrenzte Speicherkapazität ergeben. Auch bei diesem Datenaufnahmesystem werden die *list-mode*-Daten auf ein Magnetband kopiert.
Die *list-mode*-Daten werden jedoch aus zwei Gründen nicht *event* für *event* auf das Band gespeichert,

* erstens, weil die kleinste Dateneinheit beim Beschreiben und Lesen des Bandes ein Block ist, wobei unter RT11-SJ 1 Block = 512 Byte sind, und
* zweitens, weil die Effizienz des Streamertapes mit der Anzahl der zu kopierenden Blöcke wächst.

Die *list-mode*-Daten werden nun, wie aus der Abbildung 3.3 entnommen werden kann, in drei Stufen auf ein Band gespeichert.

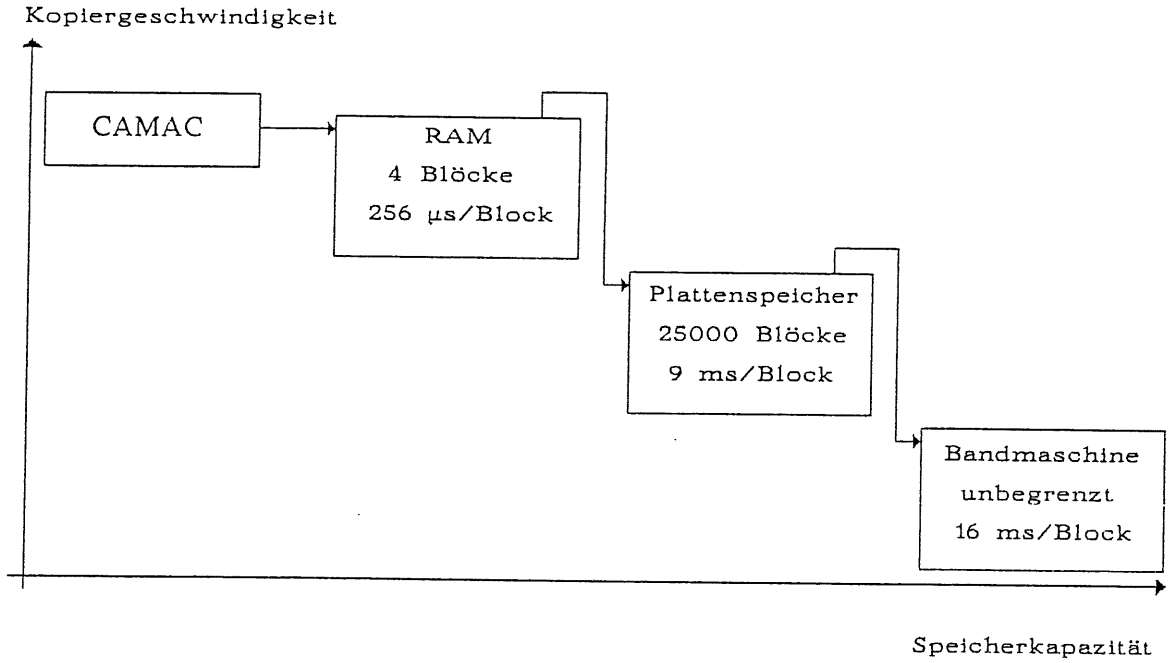


Abbildung 3.3 Speicherung der *list-mode*-Daten.
Die Koordinaten sollen nur qualitativ das Zeit-Kapazität-Verhältnis der Speichermedien wiedergeben. Die angegebenen Kopierzeiten sind Mittelwerte. Sie variieren mit der Anzahl der kopierten Blöcke.

In einer ersten Stufe werden die *event*-Daten in das RAM abgelegt. Dieser Datentransfer ist im Vergleich zu den folgenden am schnellsten. Wenn der Speicherraum von 2 KByte erschöpft ist, werden die Daten auf eine Festplatte kopiert. In dieser zweiten Stufe werden alle Daten eines *runs* gesammelt. Der dafür vorgesehene Platz wurde der Bandkapazität angepaßt und beträgt in der jetzigen Version 25000 Blöcke bei einer Bandlänge von 2400 ft[2]. Das entspricht einer Anzahl von *events* zwischen 106 000 und 3 200 000, je nach dem, ob pro *event* 56 oder nur 1 ADC ausgelesen und gespeichert wird.
In der letzten Stufe werden die Daten eines *runs* auf ein Band kopiert. Während die Stufen 1 und 2 in einem Programm Platz finden, mußte die dritte Stufe mit einem separaten Programm (COPYDT) realisiert werden.

3.3 Die Benutzerprogramme

In der Tabelle 3.1 werden diese Programme zusammengefaßt. Für die Beschreibung dieser Software, soweit sie für die Benutzung notwendig ist, wird hier global für die nachfolgenden Abschnitte dieses Kapitels auf den Teil II verwiesen. In den kommenden Abschnitten werden ergänzende Angaben zu Teilaufgaben des Systems und ihrer Umsetzung gemacht.
Der letzte Abschnitt des Kapitels befaßt sich dann mit der Ausbaumöglichkeit dieser Software.

2) Die Kapazität eines Magnetbandes hängt außer von der Länge von der Schreibdichte und der Blockgröße ab. Die Schreibdichte beträgt 1600 bpi (bits per inch). Da zwischen den Blöcken sogenannte *interrecordgaps* geschrieben werden, die eine feste physikalische Länge von 2/3 inch haben, läßt sich die Kapazität durch größere Blöcke steigern. In dieser Version wird eine Standard Blockgröße von 512 Byte pro Block verwendet. Dadurch können die Bänder problemlos auf anderen Rechenanlagen ausgelesen werden.

Programm	Funktion
INIFIL	Initialisierung der temporären *list-mode*-Datendatei auf einer Festplatte.
CALIN	Kalibration der ADCs einschließlich der vorgeschalteten Elektronik.
DAQIN	Aufnahme experimenteller *run*-Daten.
COPYDT	Kopieren der *list-mode*-Daten von der Festplatte auf ein Magnetband.
CMPTWD	*Verify* der *list-mode*-Daten auf dem Band.
COPYTD	Kopiert die *list-mode*-Daten vom Band zurück auf die Festplatte.
CALOUT	Auswertung der Daten von CALIN.
DAQOUT	Auswertung der Daten von DAQIN.
GIL	Spektrenmanipulation wie Gaußfit, Integral und Lupe.
SAVEPAS	Sichern und Laden von Parametern und Spektren.

Tabelle 3.1: Die Programme

3.3.1 INIFIL

Im Kapitel 3.2 wurde die Speicherung der *list-mode*-Daten in drei Stufen erläutert. Für die zweite Stufe ist es notwendig, daß eine temporäre Datei auf einer Festplatte eröffnet wird. Diese Initialisierung wird mit INIFIL vorgenommen. Die temporäre Datei trägt den Namen A37LST.TMP, und sie hat einen Umfang von 25000 Blöcken. Diese Datei bleibt für alle *runs* bestehen. Mit Hilfe eines Blockzählers ist zur jeder Zeit bekannt, welche der 25000 Blöcke gültige *run*-Daten beinhalten. Wenn ein *run* gelöscht werden soll, so wird nur der Blockzähler auf 0 gesetzt. Die alten Daten werden so mit den neuen Daten überschrieben. Auch beim späteren Transfer auf ein Band werden natürlich nur die gültige und nicht etwa alle 25000 Blöcke kopiert.
Die Besonderheit dieser Methode ist, daß nur **eine** temporäre Datei für **alle** *runs* eingerichtet wird. Die Alternativlösung, für **jeden** *run* eine temporäre Datei neu zu öffnen, würde das System für den Benutzer sehr unkomfortabel erscheinen lassen, was durch zwei Merkmale des Betriebssystems RT11-SJ bedingt ist:
Wenn unter RT11-SJ eine Datei auf einer Festplatte neu eröffnet wird, so sucht das Betriebssystem dafür den größten freien Speicherplatz aus. Wird diese Datei wieder gelöscht, so bleiben die Start- und Endmarkierungen der Datei bestehen, d.h., die zuvor vorhandene freie Speicherplatzgröße wird nicht mehr erreicht. Häufiges Öffnen und Löschen von Dateien zerlegen so den freien Raum in viele kleinere Räume.
Das zweite Merkmal ist, daß die angelegten Dateien *contiguous* sein müssen, d.h., die Datei muß vollständig in einem freien Speicherraum Platz finden und darf nicht in mehrere Räume verteilt werden.
Auf die obengenannte Aufgabenstellung übertragen bedeutete dies, daß nach wenigen *runs* der Festplattenspeicher derart partitioniert wäre, daß kein größerer *run* mehr möglich wäre. Eine Gegenmaßnahme, ein Komprimieren der Daten, wäre in dem Fall notwendig. Die dafür vorgesehene Betriebssystemroutine (SQUEEZE) benötigt 3-6 min.

3.3.2 CALIN

Mit diesem Programm können die elektronischen Kanäle kalibriert werden. Ein elektronischer Kanal bezeichnet die Verkettung

Detektor - Vorverstärker - Hauptverstärker - ADC.

Das Ergebnis der Kalibration soll die eindeutige Zuordung von Energiewerten zu ADC-Ausgabewerten sein. Da die ADC-Ausgabewerte ganzzahlig sind, spricht man auch von ADC-Kanäle (ADC-*channels*).
Die Kalibrationsmethode ist gezielt auf die Verwendung von Halbleiterdetektoren abgestimmt. Es werden sogenannte Kammspektren aufgenommen, d.h., mit einem energiegeeichten System bestehend aus einem Testpulser und einem externen Chargeterminator werden mehrere *peaks* bei unterschiedlichen Testpulsereinstellungen aufgefahren. Auf diese Weise werden Stützpunkte der Kalibrationskurve eines elektronischen Kanals ermittelt. Abbildung 3.4 zeigt eine typische Kalibrationskurve, die während eines Experiments am Isochronzyklotron aufgenommen wurde. (siehe Kap. 7)

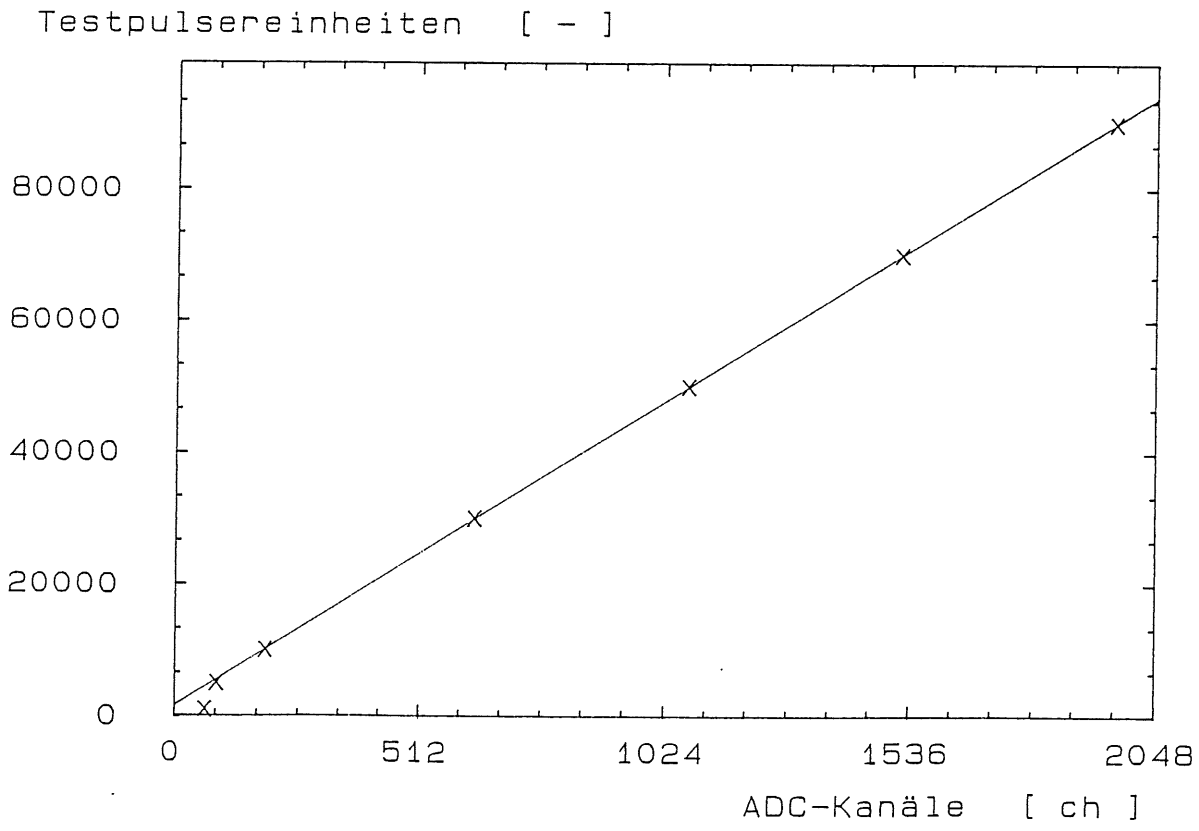


Abbildung 3.4: Kalibrationskurve eines elektronischen Kanals

Innerhalb der Kalibrationsaufgabe ergeben sich im wesentlichen zwei Problemstellungen.

* Erstens müssen für jede Kalibrationskurve die Koeffizienten für eine lineare Anpassung bestimmt werden. Diese Koeffizienten dienen dann der Energieeichung für die *online*-Auswertung. Die lineare Energieeichung ist eine Kompromißlösung. Wie aus der Abbildung 3.4 zu entnehmen ist, hat eine solche Anpasssung nur für den oberen Teil der Kalibrationskurve Gültigkeit. Eine bessere Anpassung hingegen benötigte mehr Koeffizienten und mehr Zeit in der *online*-Auswertung. Deshalb unter anderem die zweite Forderung:
* Die *list-mode*-Daten müssen in so eine Form abgespeichert werden, daß eine spätere *offline*-Auswertung möglich ist, mit der auch der nichtlineare Teil der Kalibrationskurve angepaßt werden kann. Hierzu werden sogenannte Informationsblöcke zwischen den *list-mode*-Daten der einzelnen *peaks* geschrieben. Durch sie werden gleichzeitig zwei Aufgaben erfüllt.
  * Erstens können die *peaks* im nachhinein getrennt werden, auch wenn sie im Spektrum eng benachbart stehen oder sich sogar überlappen.
  * Zweitens können alle wichtigen *peak*-Daten wie z.B. Sollenergie, Testpulsereinstellung, elektronische Kanalnummer etc. in diesem Informationsblock für das Auswerteprogramm abgespeichert werden.

Die Datenstruktur der *list-mode*-Datendatei auf einem Magnetband einschließlich der Informationsblöcke wird im Anhang A dargelegt.
Die Aufgabenlösungen werden in dem Programm CALIN als Bestandteile eines Menüs realisiert. Anhand des Flußdiagramms (Abb. 3.5) soll die Konstruktion des Programms deutlich gemacht werden. Sie besteht im wesentlichen aus zwei Teilen.

* In einem Initialisierungteil werden alle Konstanten und Variablen mit definierten Werten besetzt. Teils werden diese Werte aus einer Parameterdatei gelesen, teils werden sie interaktiv von dem Benutzer bestimmt.
* Die Kommandoschleife bildet den zweiten Teil. Sie beinhaltet die CAMAC-Routine, die in der Abbildung 3.6 detailierter gezeigt wird. Die Kommandoschleife stellt deshalb den zeitkritischen Teil des Programms dar und ist in Macro Assembler geschrieben.
  Als höchste Priorität wird als erstes nach einem LAM-Signal gefragt. Die Behandlung des LAM-Signals wurde oben im Abschnitt 3.1 erläutert.
  Wenn kein *event* zu verarbeiten ist, wird getestet, ob und wenn, welches Kommando durch die Konsole vom Benutzer eingegeben worden ist. Wenn kein oder ein unbekannter Befehl eingegeben wurde, wird der Rest der Schleife in ca. 200 µs durchlaufen und wieder an den Anfang gesprungen. Ein entgültiges Verlassen der Schleife kann nur durch den Menüpunkt "E" erreicht werden. Die Befehle werden in der Abbildung 3.5 nur knapp kommentiert. Eine ausführliche Darstellung findet sich im Teil II, Kapitel 1.2.

Wenn ein *event* zu verarbeiten ist, dann wird die CAMAC-Routine durchlaufen. Die CAMAC-Befehle am Anfang und am Ende der Routine sind aus der Abbildung 3.2 schon bekannt. Die "programmspezifischen Operationen" beginnen danach mit dem Lesen des ADC-Datenwortes. Da immer nur ein ADC zur Zeit kalibriert werden kann, wird auch nur das betreffende ADC ausgelesen. Das ADC-Datenwort wird zunächst im *list-mode* in das RAM abgelegt. Danach wird ein entsprechender Eintrag in den *displaybuffer* vorgenommen.
Ein *displaybuffer* ist ein Speicher, der die Informationen für ein Monitorspektrum beinhaltet. Die Anzahl der Speicherzellen hängt von der Auflösung des Monitorspektrums ab. Der Inhalt der Speicherzellen entspricht der Anzahl der Einträge in den jeweiligen Energiekanal. Der *displaybuffer* wird nach jedem *event* aktualisiert, auch wenn das entsprechende Spektrum nicht auf dem Graphikterminal dargestellt wird. In dem Programm CALIN ist die Größe des *displaybuffers* fest auf 2048 Speicherworte voreingestellt. Sie ist damit für ADCs mit einer maximalen Auflösung von 11 Bits eingerichtet. Wenn ADCs mit einer höheren Auflösung verwendet werden sollen, so kann der Initialisierungsteil von CALIN entsprechend geändert werden. Zu den Verwaltungsaufgaben eines *displaybuffers* gehört auch für die Bildung des Mittel- und der rms-Wertes die Aktualisierung der Größen N, $\sum x_i$, $\sum x_i^2$, wobei N die Anzahl der *events* und x_i der ausgelesen ADC-Wert ist.
Wenn das *lifedisplay* aktiv ist, d.h., daß während der Datenaufnahme der *displaybuffer* auf dem Graphikterminal dargestellt wird, muß der neue Eintrag in den *displaybuffer* auch auf dem Monitor sichtbar gemacht werden. Wenn der gewählte Maßstab nicht linear ist, muß für die Berechnung der Pixeladresse zusätzlich der Logarithmus eines Wertes gebildet werden. Hierzu wurde aus zeitkritischen Gründen keine Standard-Fortran-Routine verwendet, sondern die Berechnung wurde in Macro Assembler programmiert. Diese Assembler-Routine ist um 18.7 % schneller als ihre Alternative.

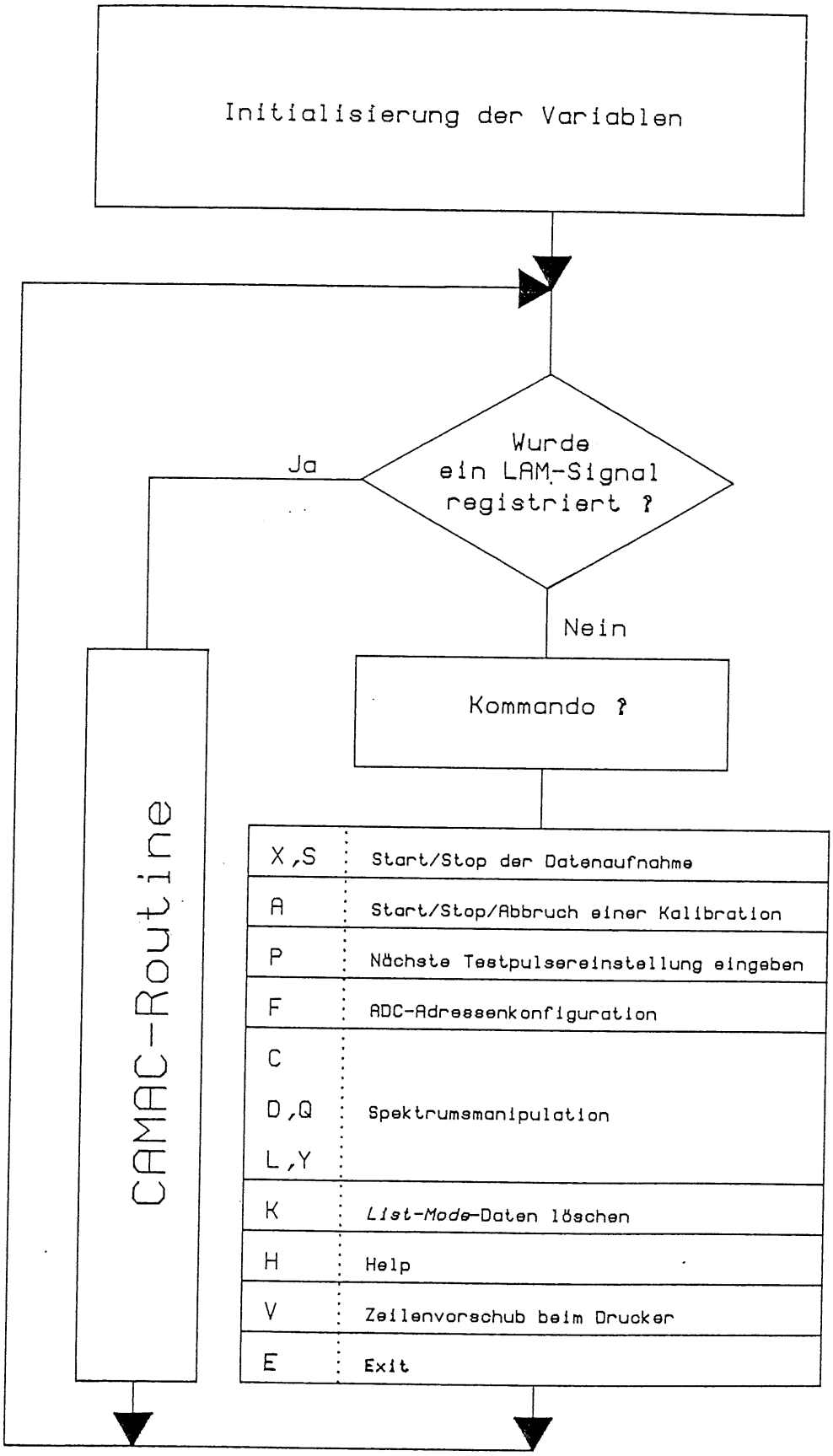


Abbildung 3.5: Flußdiagramm des Programms CALIN.

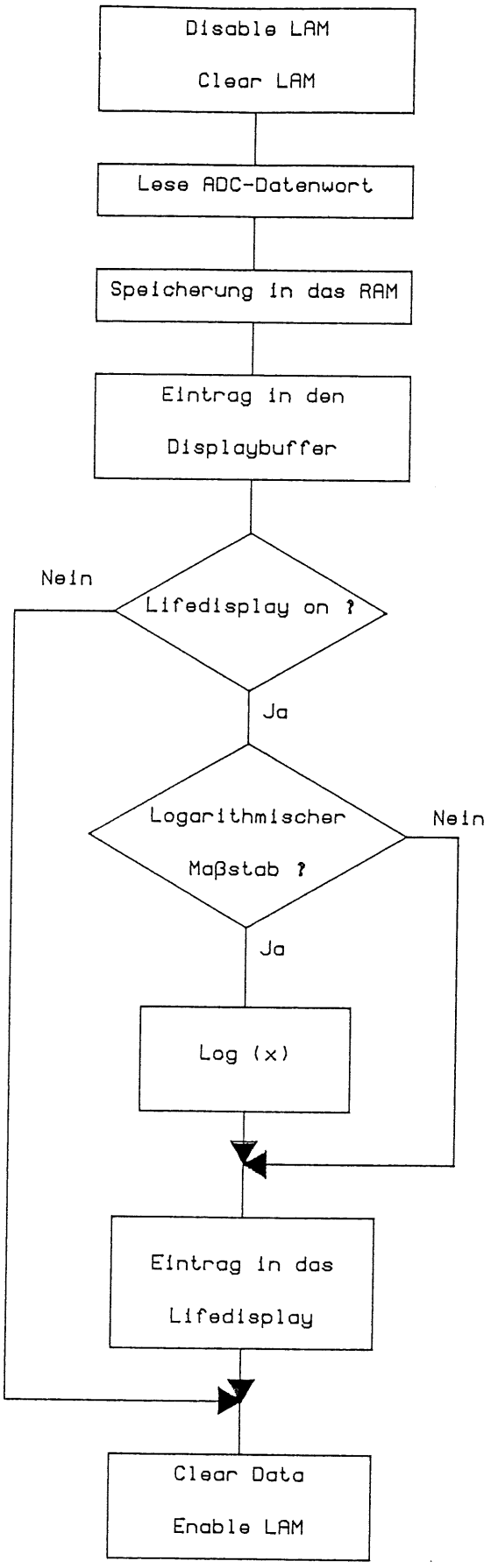


Abbildung 3.6: Die CAMAC-Routine von CALIN

3.3.3 DAQIN

Dieses Programm bildet den Kern der Software des DAQ-Systems. Mit ihm werden die experimentellen Daten aufgenommen. Die Struktur des Programms ist der des Programms CALIN sehr ähnlich und kann in der Abbildung 3.5 abgelesen werden. Entsprechend dem Aufgabenfeld ist aber der Befehlssatz ein anderer. Er wird im Teil II, Kapitel 1.1 ausführlich beschrieben. Die CAMAC-Routinen von CALIN und DAQIN sind sehr unterschiedlich. Es lassen sich vier Unterschiede feststellen (vgl. Abbildung 3.6 und 3.7).

* Pro *event* können bis zu 56 ADCs ausgelesen werden. Sie müssen im *list-mode* in das RAM gelegt werden.
* Die ausgelesenen Daten werden kalibriert, aufsummiert und in den sogenannten *sumbuffer* eingetragen. Der *sumbuffer* hat eine feste Größe von 1024 Kanälen.
* Während bei CALIN nur ein *displaybuffer* ausgelesen wird und nur ein *displaybuffer* verwaltet wird, können in DAQIN bis zu 56 *displaybuffer* angelegt werden. Die Menge der darzustellenden ADCs muß nicht mit der Menge der ausgelesenen ADCs identisch sein, so daß dem Benutzer keinerlei Einschränkungen in dieser Hinsicht auferlegt werden. Wenn in DAQIN geeichte ADCs ausgelesen werden, dann werden die Werte entsprechend kalibriert, bevor sie in einen *displaybuffer* eingetragen werden, anderenfalls unterbleibt eine Kalibration.
* Wenn das *lifedisplay* aktiv ist, muß differenziert werden, welcher *buffer* zur Zeit auf dem Graphikterminal angezeigt wird.

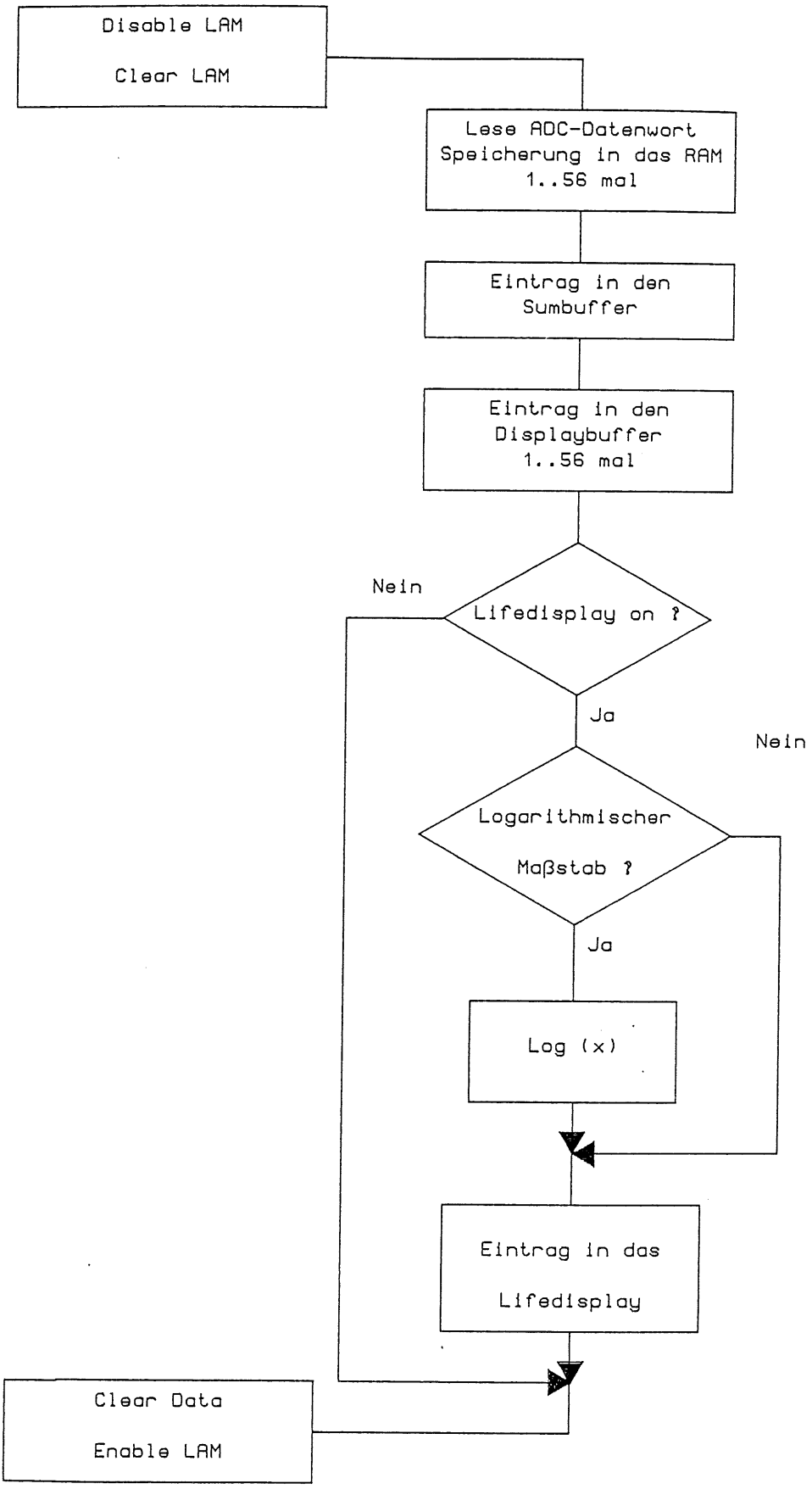


Abbildung 3.7: CAMAC-Routine in dem Programm DAQIN.

3.3.4 COPYDT und CMPTWD

Bei diesen Programmen müssen vom Benutzer keine Eingaben gemacht werden. Der Name des *runs* ist in der Parameterdatei verzeichnet und die zu kopierenden *list-mode*-Daten stehen in der temporären Datei. Das Kopierprogramm COPYDT ist speziell der Besonderheit einer Streamer-Tape-Unit angepaßt. Da diese Unit nur bei einer bestimmten Bandgeschwindigkeit schreibt und liest wird, und weil die Endgeschwindigkeit nicht nach einigen Zentimetern sondern nach erst nach einigen Dezimetern erreicht wird, hängt die durchschnittliche Schreibgeschwindigkeit auch davon ab, wie häufig das Band beim Kopieren gestoppt werden muß. Das Band muß immer dann angehalten werden, wenn das Beschreiben des *tapes* schneller ist als die Versorgung mit neuen Datenblöcken. Dieses *timing*-Problem kann dadurch minimiert werden, indem parallele Prozesse zeitlich voll ausgenutzt werden.
Dazu werden die Daten in zwei Puffer transferiert, die jeweils einen Umfang von 32 Blöcken haben. Während nun der Puffer A auf das Band geschrieben wird, wird gleichzeitig der Puffer B mit neuen Daten von der Festplatte beschrieben. Danach wird der Puffer B geschrieben, während der Puffer A neu geladen wird, usw.

Da während des Schreibens mit dem Streamertape ein *Verify* [4] der Daten nicht möglich ist, kann dies durch das Programm CMPTWD nachgeholt werden. Da außerdem mit dem Streamertape ein Rückwärtslesen nicht möglich ist, muß das Band für ein *Verify* an den Anfang gespult werden. Die Spulzeit beträgt bei einem 2400 ft langen Magnetband vom Ende bis zum Anfang ca. 2 min 40. Anschließend wird das Band vorwärts lesend auf den letzten *run* positioniert, und dann werden die Magnetband-Daten mit den noch auf der Festplatte vorhandenen Daten verglichen. Das Ergebnis dieses Vergleichs wird dem Benutzer mitgeteilt.
Die Erfahrung hat jedoch gezeigt, daß ein *Verify* nicht nach jedem *run* notwendig ist, sondern daß die Anwendung nur in Ausnahmefällen angezeigt ist.

3.3.5 COPYTD

COPYTD in das Gegenstück zu COPYDT, denn es kopiert *list-mode*-Daten vom Magnetband auf die Festplatte. Aber es können hier keine 32 Block große Puffer verwendet werden, weil die Anzahl der zu kopierenden Blöcke zu Beginn des Kopiervorgangs nicht bekannt ist. Dadurch muß das Ende der Datei blockweise "ertastet" werden.

4) *Verify* bedeutet zu verifizieren, daß die Daten fehlerfrei von der Festplatte auf das Band kopiert worden sind.

3.3.6 CALOUT und DAQOUT

CALOUT und DAQOUT sind Datenauswerteprogramme und stellen die Pendants zu CALIN und DAQIN dar. Die *list-mode*-Daten werden in CALOUT und DAQOUT *event*-weise in *displaybuffer* eingelesen, so daß sie in der gleichen Weise zur Verfügung stehen wie in den Datenaufnahmeprogrammen. Jedoch gibt es zwei wesentliche Unterschiede:

* Bei der Eichung wird eine eventuell vorhandene Nicht-Linearität der ADCs im unteren Bereich explizit durch eine kubische Kalibrationskurve berücksichtig. Dies ist notwendig, wenn relativ niedrige Signale korrekt gemessen werden solle.
* Es kann für jedes Spektrum, insbesondere für das Summenspektrum, ein logischer Trigger gesetzt werden. Das ist ein Energiefenster, und es werden nur solche *events* bearbeitet, deren Energie innerhalb dieses Fensters liegt.

3.3.7 GIL und SAVPAS

Diese Programme stellen Hilfen für den Benutzer dar.
Mit dem Programm GIL können Spektrumsmanipulationen wie Gaußfit, Integral und Lupe ausgeführt werden.

* Der Gaußfit basiert auf einer quadratischen Regression von Punkten im logarithmischen Maßstab (3 Parameterfit). Das Intervall auf der X-Achse, in dem gefittet werden soll, kann durch Marken gesetzt werden. Das Ergebnis des Gaußfits sind Mittelwert, Standardabweichung und ein Skalenfaktor.
* Die Integralfunktion gibt nicht nur den Flächeninhalt (Anzahl der *events*) eines Bereiches wieder sondern auch die Schwerpunktachse (Mittelwert) und den root-mean-square-Wert.
* Mit der Lupenfunktion können Ausschnitte vergrößert werden.

Mit SAVPAS können Parameter- und Spektrendateien gesichert und wieder geladen werden. Im Kapitel 2.2 wurde erwähnt, daß zum Informationsaustausch zwischen den Programmen Dateien angelegt werden müssen. Die Software des DAQ-Systems bedient sich drei solcher Dateien.

* A37LST.TMP, die temporäre *list-mode*-Datendatei.
* DISPLY.DAT, eine Parameterdatei.
* IBUFF.DAT, eine Datei, die alle *displaybuffer* und den *sumbuffer* beinhaltet.

Die Struktur dieser drei Dateien wird im Anhang A aufgezeigt. Die letzten beiden Dateien können nun in neue Dateien mit anderen Namen auf der Festplatte kopiert werden. Zur Zeit steht für über 400 Kopien Speicherraum zur Verfügung. Aber auch wenn dieser Raum erschöpft sein sollte, könnte neuer Raum geschaffen werden, indem Kopien auf ein Magnetband ausgelagert würden.

3.4 Ausbaumöglichkeiten der Datenaufnahmeprogramme

Durch die Anwendung der Overlay-Technik sind die Programme modular aufgebaut, so daß Erweiterungen durch zusätzliche Module, soweit der residente Teil des Programms nicht größer als 50 kByte wird, möglich sind.
Ein Ausbau von derzeit 56 *displaybuffer* auf eine höhere Anzahl ist nur auf Kosten der Auflösung der *displaybuffer* möglich, weil der Speicherplatz für alle *buffer* nicht größer als 30 KByte sein kann.
Eine Vergrößerung der Anzahl von ADCs, die im *list-mode* gespeichert werden, ist von 56 auf einige hundert ADCs möglich, wenn eine neutrale Kalibration der ADC-Werte für die Berechnung der Summe für den *sumbuffer* ausreichend ist.

Kapitel 4

Steuerung der Detektorspannungen

Außer der Datenaufnahme und der Datenauswertung ist noch vorgesehen, mit dem DAQ-System die Spannungsversorgung der Detektoren zu regeln. Das Projekt ist noch nicht abgeschlossen, da eine Komponente noch in der Fertigstellung ist (siehe unten). Die Aufgaben der Spannungsregelung und ihre Lösung sind die Themen der folgenden zwei Abschnitte.

4.1 Problematik der Spannungsregelung

Ein geladenes Teilchen, das durch einen Oberflächensperrschicht-Detektor defundiert, erzeugt auf seinem Weg Elektron-Loch-Paare. Damit nun alle Elektron-Loch-Paare erfaßt werden können, muß ein elektrisches Feld den Detektor vollständig durchziehen, das die Elektronen und Löcher trennt und an den Elektroden sammelt. Eine solche Ladungsbewegung kann dann außerhalb des Detektors als Stromfluß gemessen werden.
Zum Aufbau eines elektrischen Feldes wird an den Detektor eine Spannung U in Sperrichtung angelegt. In erster Näherung kann das entstehende Feld als das eines Plattenkondensators betrachtet werden, dessen Plattenabstand d jedoch nicht konstant ist, sondern mit der angelegten Spannung variiert. Im unteren Spannungsbereich wächst $d \sim \sqrt{U}$. Wenn d die Dicke des Detektors erreicht hat, verändert es sich bei weiterer Erhöhung der Spannung nicht mehr. Dieser obere Spannungsbereich, in dem d maximal ist, wird im allgemeinen als Arbeitsbereich für Oberflächensperrschicht-Detektoren gewählt.
Damit der obengenannte Stromfluß auch gemessen werden kann, wird die Spannung über einen Vorwiderstand angelegt. Dieser Vorwiderstand ist ein Bestandteil eines Vorverstärkers (*pre–amplifier*) und wird deshalb hier R_{PA} genannt. Eine Spannungsversorgung über einen Vorwiderstand wird in der Abbildung 4.1 gezeigt.

Bei einem Einsatz der Detektoren in einem Experiment müssen nun die Detektorspannungen U_D eingestellt und ständig überwacht werden, da der Sperrstromwiderstand R_D unter anderem auch von der Temperatur abhängig ist. Erfahrungen aus der Testmessung am Isochronzyklotron zeigen, daß das manuelle Einstellen einer individuellen Spannung U_D für beispielsweise 8 Detektoren ca. 45 min benötigt. Der Grund für die verhältnismäßig lange Einstellzeit wird im folgenden beschrieben.
Das SY 127 der Firma C.A.E.N. ist ein Hochspannungs-Versorgungsgerät (Abbildung 4.1), das bis zu 40 Einzeldetektoren speisen kann, und das über ein Terminal programmierbar ist. Da nun der Sperrstromwiderstand des Detektors R_D im allgemeinen nicht bekannt ist, muß die Spannung U_D mittels des Stromes I berechnet werden.

$$U_D = U_0 - I * R_{PA} \qquad \text{Gl. (4.1)}$$

Der Strom I und die Spannung U_0 werden vom SY 127 gemessen und angezeigt. Die Spannung U_0 wird dabei auf ± 0.2 V genau angegeben, was für die meisten Anwendungen ausreichend ist. Der Strom wird auf $\pm 0.2\,\mu$A genau angezeigt. Das ist nicht ausreichend, weil eine Einstellung der Dektektorspannung mit

$$\pm 0.2\,\mu A * 100\ M\Omega = \pm 20\ V$$

nicht akzeptabel ist. Aus diesem Grund muß der Strom separat gemessen werden. Eine solche Meßanordnung ist in der Abbildung 4.2 zu sehen.

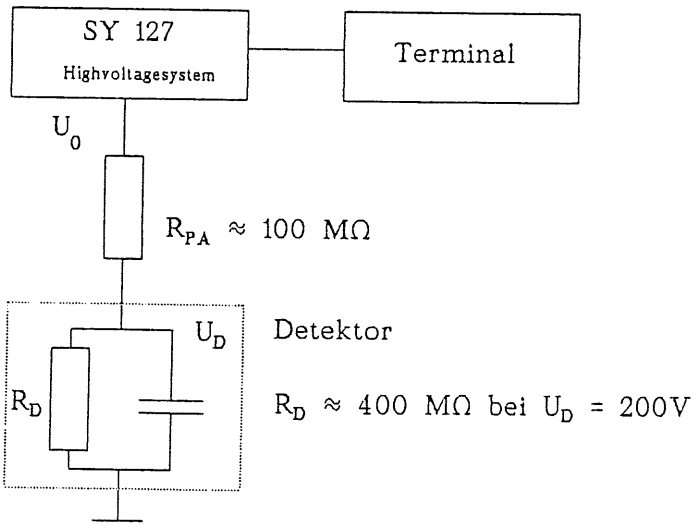


Abbildung 4.1: Spannungsversorgung eines Detektors. Das elektronische Ersatzbild der verwendeten Oberflächensperrschicht-Detektoren ist in erster Näherung eine Parallelschaltung aus einem Kondensator und einem Widerstand. U_0 ist die Ausgangsspannung des SY 127, U_D ist die am Detektor anliegende Spannung.

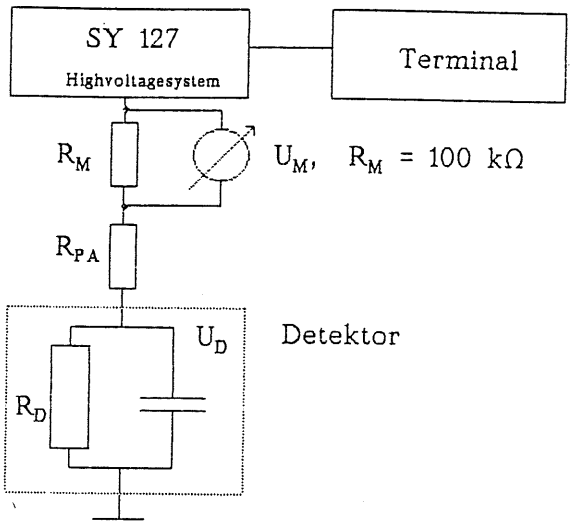


Abbildung 4.2: Spannungsversorgung mit separater Strommessung.

$$I = U_M / R_M$$

Gl. (4.1) muß entsprechend korrigiert werden zu

$$U_D = U_0 - I * (R_{PA} + R_M). \qquad \text{Gl. (4.2)}$$

Eine bestimmte Spannung U_D wird nun durch sukzessive Approximation erreicht. Der Algorithmus zum Einstellen einer Detektorspannung $U_{D,ist} = U_{D,soll}$ ist dann der folgende:

* U_0 anlegen. (Am Anfang wird $U_0 = U_{D,soll}$ gewählt.)
* I messen.
* Aus Gl. (4.2) $U_{D,ist}$ berechnen.
  Wenn $U_{D,soll}$ ausreichend[1)] gut getroffen wurde, dann wird die Approximation abgebrochen, anderenfalls wird R_D bestimmt und ein neuer Wert für U_0 festgelegt:

$$U_0 = \left(\frac{U_{D,soll}}{R_D} \right) * (R_M + R_{PA} + R_D)$$

* zurück zu Punkt 1.

Die manuelle Ausführung dieser Prozedur benötigt viel Zeit. Mit der im nächsten Abschnitt beschriebenen Automatisierung ist nicht nur die gewünschte Sollspannung an einem Detektor schneller einstellbar, sondern es ist nun auch die Aufnahme einer ganzen Diodenkennlinie eines oder mehrerer Detektoren praktizierbar.

1) "Ausreichend" bedeutet in diesem Zusammenhang die maximal erreichbare Genauigkeit einschließlich einer individuell vorgegebenen Toleranz.

4.2 Rechnergesteuerte Spannungsversorgung

Der obengenannte Algorithmus kann in die Form eines Programms umgesetzt werden. Dazu müssen hardware-seitig die Voraussetzungen für drei Aufgaben geschafft werden:

* Erstens, die Spannung U_0 muß am SY 127 "abgelesen" werden.
* Zweitens, diese Spannung muß durch Programmieren des SY 127 verändert werden können und
* drittens, der Strom I muß gemessen werden.

Eine Realisierung dieser Anforderungen wird in Abbildung 4.3 gezeigt.

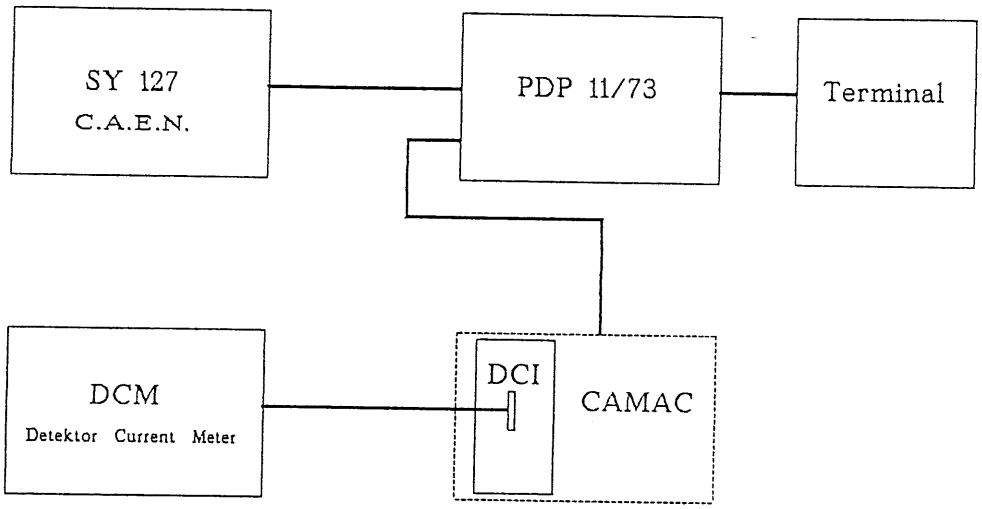


Abbildung 4.3: Erweiterung des DAQ-Systems mit dem SY 127 und dem DCM.

An das SY 127 ist statt eines Terminals die PDP 11/73 über eine serielle Schnittstelle angeschlossen.
Mit einem "Terminalsimulations"-Programm[2)] ist auch auf Software-Ebene die Verbindung zu dem SY 127 hergestellt. Damit sind die notwendigen Voraussetzungen für die Regelung und Überwachung der Spannungsversorgung für die Detektoren vorhanden.
Für die Strommessung wurden das DCM (Detektor Current Meter; ist zur Zeit noch in der Bauphase) und das DCI (Detektor Current Interface) entwickelt. In den nächsten drei Abschnitten werden die beiden Einheiten und ihre Ausbaufähigkeit vorgestellt.

2) Dieses Programm, es trägt den Namen CAEN, simuliert für das SY 127 ein Terminal, d.h., die Befehle werden per Konsole an das SY 127 gegeben, und das Echo erscheint auf dem Monitor.

4.2.1 Das Detektor Current Meter (DCM)

Wie auch bei der manuelle Messung wird mit dem DCM im Prinzip der Strom indirekt durch den Spannungsabfall an einem Meßwiderstand R_M gemessen, wie aus Abbildung 4.4 hervorgeht.

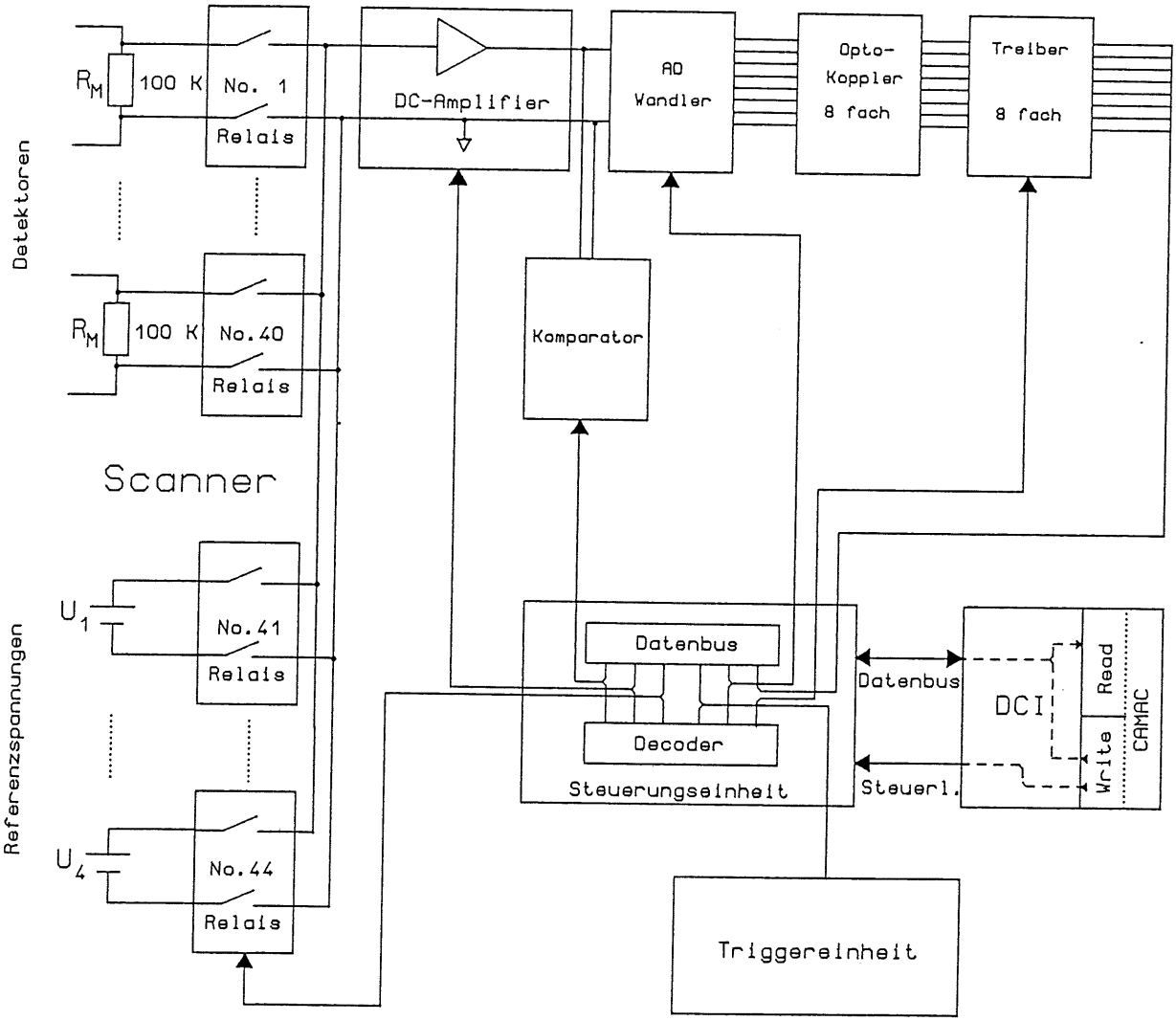


Abbildung 4.4: Schematischer Aufbau des DCM. Das Zeichen "▽" symbolisiert die erdfreie Masse.

Da der Meßwiderstand direkt hinter der Spannungsversorgung liegt, ist sein Potential hoch gegenüber dem Erdpotential (bis 1000V). Daher muß ein Teil der Elektronik eine erdfreie Masse haben. An drei Bauteilen bzw. Bauteilgruppen muß diese Isolation gewährleistet werden.

* An den Relais (es wurden Reed-Relais verwendet, die sich insbesondere durch ihre kompakte Bauform und durch ihren niedrigen Spulenstrom auszeichnen),
* an den Optokopplern und
* an dem Netztransformator, der die Spannungsversorgung für den Gleichspannungsverstärker (DC-Amplifier) und für den Analog-Digital-Wandler (AD-Wandler) leistet.

Erläuterung der einzelnen Baugruppen des DCM:

Der Scanner

Mittels Reed-Relais kann jeweils einer von 40 Meßwiderständen oder eine der 4 Referenzspannungen angewählt (*scan*) werden.

Der Gleichspannungsverstärker

Der Gleichspannungsverstärker wurde nach Vorlage des Digital-Multi-Meter Model 177 der Firma Keithly aufgebaut. Seine Verstärkung ist mit Reed-Relais in drei Stufen regelbar (1x, 10x, 100x). Die Feineinstellung des Verstärkungsfaktors kann auf zwei Wegen erfolgen.

* Durch Trimm-Potentiometer.
* Eine Korrektur der Feineinstellung ist auch durch die Software des Anwenders vorgesehen, da die Elektronik termischen Schwankungen unterliegt. Hierfür sind 4 Referenzspannungen (2V, 200mV, 20mV, 0V) eingerichtet, die durch Reed-Relais angewählt werden können. Die Referenzspannungen sind sehr temperatur- und langzeitstabil und können zusätzlich an Buchsen außerhalb des Gehäuses abgegriffen werden, um geringfügige Veränderungen in der Software zu berücksichtigen.

Der Meßbereich des DCM kann in der Tabelle 4.1 abgelesen werden.

Der Komparator

Mit dem Komparator kann festgestellt werden, ob die Ausgangsspannung des Verstärkers aus dem zulässigen Definitionsbereich des AD-Wandlers herausfällt. Das ist der Fall, wenn der Verstärker übersteuert, oder wenn der zu messende Strom eine andere Fließrichtung hat. Die Statussignale werden wegen der unterschiedlichen Potentiale durch Optokoppler zu der Steuerungseinheit übertragen.

Verstärkung	Max. zulässiger Spannungsabfall U_M am R_M	U_M/R_M	Toleranz
[-]	[mV]	[µA]	[µA]
1	2000	20.0	0.078
10	200	2.0	0.0078
100	20	0.2	0.00078

Tabelle 4.1: Die drei Meßbereiche des DCM. Die Toleranz ist durch die 8-Bit-Auflösung des AD-Wandlers bedingt und beträgt 1/256 des Maximalwertes.

Der AD-Wandler

Der AD-Wandler digitalisiert den Analogwert in ein 8-Bit-Wort. Beginn und Halt einer Konvertierung wird durch die Steuerungseinheit überwacht.

Die Treiberstufe

Die Treiberstufe hat hier keine Verstärkungsfunktion, sondern die Aufgabe eines Relais, damit die Signale des Optokopplers nicht ununterbrochen auf dem Datenbus anliegen.

Die Steuerungseinheit

Diese Einheit ist mit dem DCI über 8 Daten- und 5 Steuerleitungen verbunden. Die Datenleitungen (Datenbus) werden bidirektional benutzt, während die Steuerleitungen nur in einer Richtung betrieben werden. Das DCI sendet über die Steuerleitungen Befehlsworte. Ein Befehlswort beinhaltet kodiert die Information

* über die Richtung, in der der Datenbus geschaltet werden soll (read/write) und
* welche Baugruppe auf den Datenbus zugreifen darf.

Mit einem Beispiel soll die Funktion der Steuerungseinheit veranschaulicht werden.
Es wird angenommen, es soll das Relais Nr. 15 für eine Strommessung aktiviert werden. Dann heißt der erste Befehl: "Selektiere den Scanner". Über den Datenbus erhält der Scanner vom DCI die Nummer des Relais, das aktiviert werden soll. Alle anderen Baugruppe ignorieren indes die Signale auf dem Datenbus. Die Relaisnummer wird in einem Register gepuffert, damit es solange angezogen bleibt bis ein anderes Relais des Scanners aktiviert wird.

Für vollständige Strommessung sind weitere Befehle notwendig. Es muß geprüft werden, ob die Ausgangsspannung des Verstärkers im Definitionsbereich liegt. Dies wird mit dem Befehl erreicht: "Aktiviere den Komparator". Der Komparator sendet dann die Statussignale über den Bus zum DCI. Wenn die Signale in Ordung sind, wird der AD-Wandler durch einen dritten Befehl gestartet. Mit einem vierten Befehl kann der Status des AD-Wandlers (*ready/ busy*) über den Bus zum DCI übermittelt werden. Nach Beendigung der Konvertierung wird mit dem letzten Befehl der Treiber aktivert, durch den der digitale Meßwert zum DCI gesendet wird.

Die Trigger-Einheit

Eine zusätzliche Funktion, die nicht mit der Aufgabestellung der Strommessung zusammenhängt, ist die Verarbeitung (senden/empfangen) von externen Triggersignalen. Somit könnte z.B. ein Programm von einem externen Zähler, der Start- und Stop-Signale ausgibt, gesteuert werden.

4.2.2 Das Detektor Current Interface (DCI)

Das DCI ist ein Adapter zwischen dem DCM und dem CAMAC. Es sorgt dafür, daß die Signale des DCM nicht ununterbrochen auf den CAMAC-Bus und umgekehrt geschaltet sind. Für die Steuerung des DCI werden die CAMAC-spezifischen Schreib- und Lesebefehle verwendet.

4.2.3 Ausbaufähigkeit des DCM

Mit dem jetzigen Stand sind $2^{(8+5)} = 8192$ Einzelinformationen kodierbar. Zur Zeit wird davon nur ein Bruchteil verwendet, so daß hier genügend Reserven vorhanden sind. So könnte z.B. die Anzahl der Scannerkanäle von derzeit 44 auf 256 Kanäle erhöht werden. Ein solcher Ausbau wird jedoch durch die Gehäusegröße beschränkt. Aber mit einer wenig aufwendigen Modifikation der Hardware wäre es möglich, mehrere DCMs über ein einziges DCI auzusteuern.

Kapitel 5

Korrektheit der Programme

Dieses Kapitel ist in zwei Abschnitte gegliedert. Im ersten Teil wird der korrekte Ablauf der Programme untersucht, und im zweiten Teil wird die Rechengenauigkeit und die Fehlertoleranz der Anzeigenwerte diskutiert.

5.1 Test der Software

Die Programme wurden schon während der Entwicklung ständig auf Richtigkeit hin überprüft. Dabei wurde eine schnelle Methode, das Programm-Testen, verwendet. Bei dieser Methode wird der Programmablauf mit bestimmten Eingabewerten, für die die Ausgabewerte bekannt sind, kontrolliert. Streng genommen ist diese Methode kein Beweis für die Korrektheit eines Programms, wie Wirth feststellt [WIR78]:
"Das Prüfen von Programmen durch empirisches Testen kann höchstens die Gegenwart von Fehlern aufzeigen, niemals jedoch die Korrektheit eines Programms garantieren."
Dennoch wurde diese Test-Methode benutzt

* erstens, weil sie in der Programmentwicklung angewendet wurde, um schnell Fehler aufzudecken.
* zweitens, weil sie der Erfahrung nach zum Prüfen der Korrektheit eines Programms meistens ausreichend war, und
* drittens, weil die alternative Methode, die Programmverifikation [WIR78], die hinreichend die Richtigkeit von Programmen aufzeigen kann, bei umfangreichen Programmen, wie sie hier vorliegen, zeitlich zu aufwendig gewesen wäre.

Die durchgeführten Tests sind im Anhang B aufgelistet.

Bei diesen Tests wird auch das Verhalten von Eingaben berücksichtigt, die außerhalb des entsprechenden Definitionsbereiches liegen. Alle Fehleingaben können dabei nicht auf der Programm-Ebene erkannt und abgefangen werden. So wird z.B. auf Programm-Ebene erkannt, wenn statt einer geforderten Ziffer ein Buchstabe eingegeben wird. Dagegen führt die Eingabe von «CRTL C» [1] dazu, daß das Betriebssystem diese Eingabe interpretiert und nicht an das Programm weiterleitet.
Reaktionen dieser Art, wo Fehler nicht auf Programm-Ebene abgefangen werden können, werden im Kapitel "Trouble Shooting" im Teil II behandelt.

5.2 Fehlerabschätzung der Ausgabewerte

In diesem Abschnitt sollen die Fehler für die Ausgabewerte von *online*-Programmen (DAQIN, CALIN) und *offline*-Programmen (DAQOUT, CALOUT, GIL) abgeschätzt. werden. Da mehrere Fallunterscheidungen vorgenommen werden müssen, erhalten die einzelnen Fälle zur besseren Übersicht eigene Kapitelnummern.
Es werden nur diejenigen Ausgabewerte diskutiert, deren rechnerinterne Darstellung *floating-point* ist, weil alle anderen Werte sowohl in der Darstellung als auch in der Berechnung exakt sind. Der Rechner, die PDP 11/73, stellt *floating-point*-Werte ihrerseits auf zwei verschiedene Arten dar.

* *single-floating-point* und
* *double-floating-point.*

In den Programmen DAQIN und CALIN werden nur *single-floating-point*-Werte und in allen anderen Programmen werden beide Darstellungsformen verwendet. Es wird mit den *single-floating-point*-Werten begonnen.

5.2.1 *Single-Floating-Point*-Werte

Es handelt sich dabei um folgende Ausgabewerte:

*	<E>	Mittelwert
*	S	rms-Wert
*	Counts	Anzahl der *events*
*	Overfl.	Überlaufzähler

Für Mittel- und *root-mean-square*-Wert setzt sich die Fehlertoleranz

* aus den Rundungsfehlern in den Rechenoperationen,
* aus dem Kalibrationsfehler und
* aus der begrenzten ADC-Auflösung zusammen.

Für die Zählvariablen (Counts und Overfl.) hängt die Fehlertoleranz nur von den Rundungsfehlern der Rechenoperationen ab.
Im weiteren werden die Komponenten, aus denen sich die Fehlertoleranz zusammensetzt, behandelt. Es wird mit den Rundungsfehlern in den Rechenoperationen begonnen, und am Beispiel der Variablen "Counts" werden ihre Wirkungen demonstriert.

---

1) «CRTL C» führt in diesem Fall zur Terminierung des laufenden Programms.

5.2.1.1 Rundungsfehler

Pro *event* wird zu der Variablen "Counts" der Wert "1" addiert.

$$\text{Counts} = \sum_{i=1}^{N} 1 \qquad \text{Gl. (5.1)}$$

wobei N die Anzahl der *events* ist.
Das Ergebnis dieser Addition ist in der Praxis nicht immer exakt. Zur Erläuterung muß die rechnerinterne Darstellung eines *single-floating-point*-Wertes näher betrachtet werden:
X sei eine Zahl, die als *single-floating-point* gespeichert werden soll.
X wird dann folgendermaßen dargestellt:

$$|X| = q * 2^n \text{ , mit } n \in \mathbb{Z} \text{ und } 1 > q \geq 1/2.$$

Der Exponent n, die Mantisse q und das Vorzeichen von X werden in insgesamt 4 Bytes abgespeichert:

Vorzeichen	1 Bit
Exponent	8 Bits
Mantisse	23 Bits
	32 Bits

Nebenbei sei bemerkt, daß alle so abspeicherbaren Zahlen X eine Teilmenge von $\mathbb{Q}$ bilden.
Durch den Wertebereich von q bedingt hat das höchste Bit der Mantisse immer den Wert "1". Deshalb ist eine Speicherung dieses Bits nicht notwendig. Dafür werden die 23 niedrigeren Bits gespeichert, so daß immer 24 Bits der Mantisse bekannt sind.

Bei der obengenannten Addition tritt ein Rundungsfehler dann auf, wenn die Mantisse der Summe mehr als 24 Bits benötigt. Bei den Rechenoperationen unterscheidet der Prozessor zwei Modi:

* Im *roundmode* wird die Summe aufgerundet, wenn das 25. Bit der Mantisse den Wert "1" hat.
* Im *chopmode* wird immer abgerundet.

Die unterschiedlichen Wirkungen der Modi werden am Beispiel der Gl. (5.1) in der Abbildung 5.1 deutlich. Bis zu 2^{24} *events* liefert diese Addition exakte Ergebnisse. Für das vorgesehene Einsatzgebiet des DAQ-Systems sind keine *runs* mit mehr als 16 Mio. *events* ($2^{24} \approx 16\ 10^6$) zu erwarten, so daß für die Zählvariablen (Counts und Overfl.) keine Rundungsfehler auftreten werden.

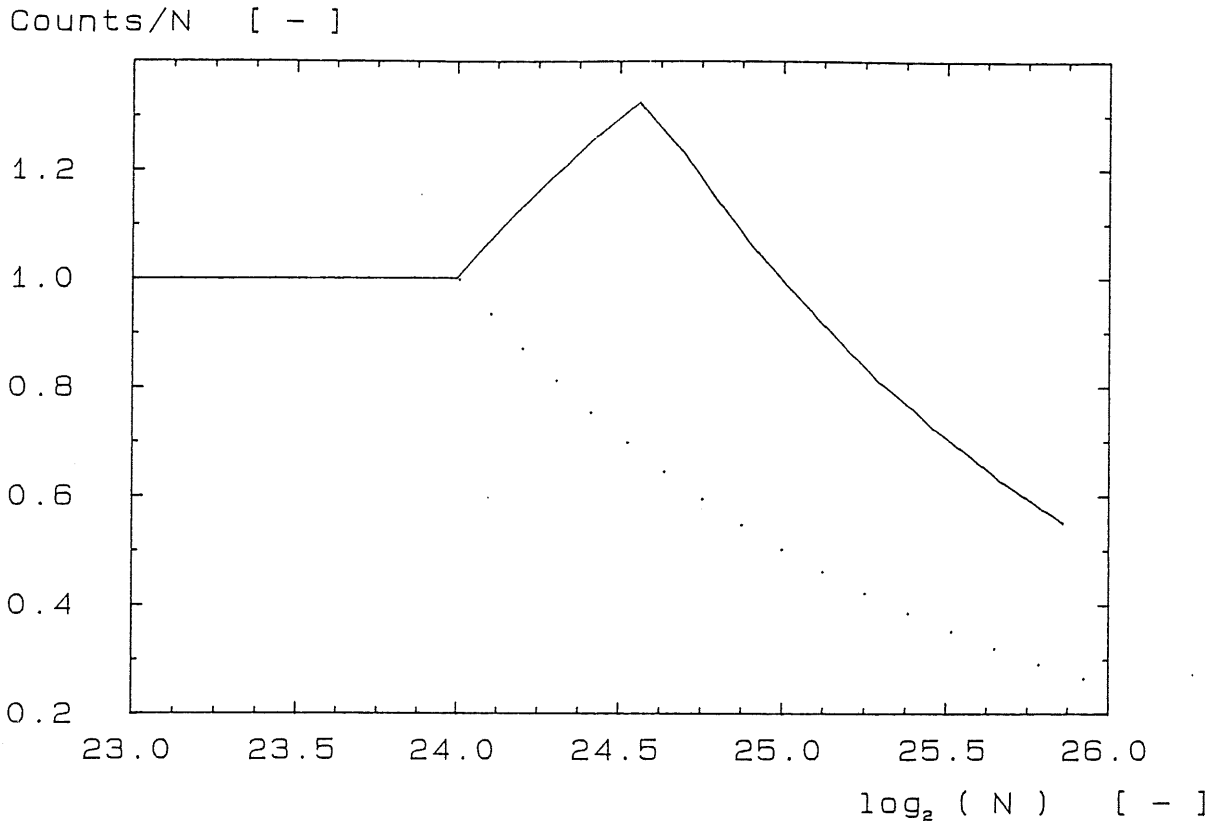


Abbildung 5.1: Rundungsfehler bei *floating-point*-Additionen.
Die durchgezogene Linie zeigt den Einfluß des *roundmodes* und die punktierte Linie den des *chopmodes*.

Im folgenden wird für den Mittelwert und die Standardabweichung die kritische Anzahl von *events* ermittelt.
Für die Berechnung des Mittelwertes wird die Summe

$$\sum_{i=1}^{N} x_i$$

gebildet. x_i ist eine Kanalnummer eines *displaybuffers*. Im ungünstigsten Fall sind alle $x_i = 2048$. Daraus folgt dann für N:

$$2^{24} = \sum_{i=1}^{N} 2048 \qquad <=> \qquad N = 8192.$$

D.h., ab 8192 *events* (bei $x_i = 2048 \; \forall \, x_i$) treten für den Mittelwert Rundungsfehler auf.

Eine analoge Überlegung wird für die Standardabweichung gemacht. Dort wird die Summe

$$\sum_{i=1}^{N} x_i^2$$

gebildet. Auch hier wird der ungünstigste Fall angenommen.
Es folgt dann für N:

$$2^{24} = \sum_{i=1}^{N} 2048^2 \quad <=> \quad N = 4.$$

Prognosen über die Größe der auftretenden Rundungsfehler können aus zwei Gründen nur schwer gemacht werden:

* Erstens liefert unter Umständen das gleiche Spektrum unterschiedliche Werte für Mittelwert und Standardabweichung, wenn die Reihenfolge der eintreffenden *events* eine andere ist.
* Zweitens hat die Form des Spektrums Einfluß auf den Rundungsfehler.

Ein Ausgabe des Fehler ist daher auch nicht möglich. In der Praxis können diese Werte jedoch **ohne** Rundungsfehler in dem Programm GIL berechnet und mit den hier angezeigten Werten verglichen werden (siehe auch Teil II). Weshalb bei GIL kein Rundungsfehler auftritt, wird weiter unten erklärt.

5.2.1.2 Kalibrationsfehler

Die Kalibrationskoeffizienten werden nur mit eingeschränkter Genauigkeit dargestellt. Daraus ergeben sich systematische Fehler bei der Energieeichung der ADC-Werte. Je nach Kalibrationsart müssen drei Fälle unterschieden werden:

* In DAQIN werden die ADC-Daten linear geeicht.
* In DAQOUT werden die Daten teils linear und teils kubisch geeicht.
* In CALIN und CALOUT entsteht trivialerweise kein Kalibrationsfehler, weil die ADC-Werte definitionsgemäß nicht energiegeeicht werden.

Sein Fehler ist in *displaybuffer*-Kanaleinheiten angegeben. Es wird dabei unterstellt, daß die *displaybuffer*-Auflösung 2048 Kanäle beträgt. Ist die tatsächliche Auflösung geringer, so ist der Fehler entsprechend kleiner.

Fall 1: DAQIN
Die Kalibrierungfunktion lautet:

$$Y = (x + b) * a ,$$

wobei x der ausgelesene ADC-Wert ist und a, b die Kalibrierungskoeffizienten sind. $\triangle a$ und $\triangle b$ seien Fehler, die bei der Berechnung von a und b entstehen. Zur Abschätzung ihres Einflusses wird das Gaußsche Fehlerfortpflanzungsgesetz angewendet.

$$\sigma_Y = \sqrt{\left\{ \left.\frac{\partial Y}{\partial b}\right|_{b_0} \Delta b \right\}^2 + \left\{ \left.\frac{\partial Y}{\partial a}\right|_{a_0} \Delta a \right\}^2}$$

$$\Rightarrow$$

$$\sigma_Y = \sqrt{\left\{ a \ \Delta b \right\}^2 + \left\{ (x + b) \ \Delta a \right\}^2} \qquad \text{Gl. (5.2)}$$

(x + b) ist eine *integer*-Operation, so daß das Resultat zwischen -32768 (= 2^{15}) und +32767 liegen kann.
Δa entspricht der Genauigkeit der *single-floating-point*-Darstellung.

$$\Delta a \leq 2^{-24} * a$$

Da b eine *integer*-Zahl[2] ist, entspricht Δb dem Fehlbetrag, der durch Rundung einer *single-floating-point*-Zahl in eine *integer*-Zahl entsteht. Somit ist

$$0 \leq \Delta b \leq 1/2.$$

Gl. (5.2) kann somit reduziert werden zu:

$$\sigma_Y \leq \sqrt{\left\{ \frac{1}{2} a \right\}^2 + \left\{ a * 2^{-9} \right\}^2} \approx \frac{1}{2} a$$

Der Hauptanteil an dem systematischen Fehler entsteht also durch die Rundung von b zu einem *integer*-Wert. a läßt sich in CALIN abfragen und liegt in der Praxis zwischen 0.2 und 5, so daß σ_Y bis zu 2.5 Kanäle betragen kann.

2) Eingangs wurde erwähnt, daß nur Werte, die als *floating-point* dargestellt werden, untersucht würden. Die Variable b stellt die einzige Ausnahme dar, weil b mit *floating-point*-Operationen berechnet wurde. Aus Speicherplatzgründen wurde b auf eine *integer*-Variable reduziert.

Fall 2: DAQOUT
Die Kalibrierungsfunktionen lauten:

$$Y = (x + b) a , \qquad \text{für } x \geq x_{Thres}$$

$$Y = c x^3 + d x^2 + e x + f , \qquad \text{für } x < x_{Thres}$$

wobei x der ausgelesene ADC-Wert ist und a ... f die Kalibrierungskoeffizienten sowie $\triangle a$ ... $\triangle f$ die dazugehörigen Fehler sind. Die Koeffizienten und der Wert x_{Thres} können im Programm CALOUT ausgegeben werden (siehe auch Teil II).

A) $\quad x \geq x_{Thres}$

$$\sigma_Y = \sqrt{ \{ a \triangle b \}^2 + \{ (x + b) \triangle a \}^2 }$$

Hier sind a und b *single-floating-point*-Werte, so daß $\triangle a$ und $\triangle b$ die Genauigkeiten der *single-floating-point*-Darstellung repräsentieren:

$$\triangle a = 2^{-24} * a$$
$$\triangle b = 2^{-24} * b$$

$$\sigma_Y \leq 2^{-24} \sqrt{ \{ a b \}^2 + \{ (x + b) a \}^2 }$$

Wenn für a, b und x ungünstige Werte wie im Fall 1 gewählt werden, folgt für σ_Y:

$$\sigma_Y \leq 2 * 10^{-5} \text{ Kanäle}$$

Dieser Fehler ist deutlich kleiner als im Fall 1, da b hier ein *single-floating-point*-Wert ist.

B) $x < x_{Thres}$

$$\sigma_Y = \sqrt{\left\{x^3\ \Delta c\right\}^2 + \left\{x^2\ \Delta d\right\}^2 + \left\{x\ \Delta e\right\}^2 + \Delta f^2}$$

Die Koeffizienten c ... f sind *single-floating-point*-Werte, so daß wie oben durch Δc ... Δf die Darstellungsgenauigkeit von 2^{-24} wiedergegeben wird. Es ergibt sich entsprechend für σ_Y:

$$\sigma_Y \leq 2^{-24} \sqrt{\left\{x^3\ c\right\}^2 + \left\{x^2\ d\right\}^2 + \left\{x\ e\right\}^2 + f^2}$$

Auch hier hängt σ_Y von den Beträgen der Koeffizienten ab und kann für jeden Einzelfall berechnet werden. Ein typischer Wert für σ_Y ist:

$$\sigma_Y \leq 10^{-4} \quad \text{Kanäle}$$

5.2.1.3 Die ADC-Auflösung

Die begrenzte ADC-Auflösung bewirkt einen **systematischen** Fehler, weil die Zuordnung der elektronischen Impulse am *line-input* eines ADCs zu ihren digitalen Ausgangswerten **eindeutig** ist.
Je nach Kalibrationsart müssen auch hier drei Fälle unterschieden werden:

* In DAQIN werden die ADC-Daten linear geeicht.
* In DAQOUT werden die Daten teils linear und teils kubisch geeicht.
* In CALIN und CALOUT werden die ADC-Daten nicht energiegeeicht.

Der Fehler wird wieder nach dem Gaußschen Fehlerfortpflanzungsgesetz berechnet und wird in *displaybuffer*-Kanaleinheiten angegeben. σ_Y wird dann zu:

$$\sigma_Y = \sqrt{\left\{\left.\frac{\partial Y}{\partial x}\right|_{x_0} \Delta x\right\}^2}$$

Fall 1: DAQIN
Für die lineare Kalibration ergibt sich dann

$$\sigma_Y = a\ \Delta x \quad , \quad \text{wobei} \quad 0 \leq \Delta x \leq 1 \text{ ist.}$$

Fall 2: DAQOUT
Für lineare Kalibration ($x \geq x_{Thres}$) ergibt sich

$$\sigma_Y = a \, \Delta x \quad , \quad \text{wobei} \quad 0 \leq \Delta x \leq 1 \text{ ist.}$$

Für $x < x_{Thres}$ wird σ_Y zu:

$$\sigma_Y = \Delta x \left\{ 3 c x_0^2 + 2 d x_0 + e \right\}$$

Für $x = x_{Thres}$ wird definitionsgemäß (siehe auch Teil II, Kap. 1.5)

$$\left\{ 3 c x_0^2 + 2 d x_0 + e \right\} = a .$$

Das σ_Y wird in dem Programm CALOUT (Menüpunkt "P") für Kalibrationskurven in Energieeinheiten berechnet (siehe auch Teil II, Kap. 1.5).

Fall 3: CALIN und CALOUT
Da in diesen Programmen nicht kalibriert wird, folgt

$$\sigma_Y = \Delta x \quad , \quad \text{wobei} \quad 0 \leq \Delta x \leq 1 \text{ ist.}$$

5.2.2 *Double-Floating-Point*-Werte

Nachdem die *single-floating-point*-Werte diskutiert worden sind, soll nun die Fehlertoleranz der *double-floating-point*-Werte beleuchtet werden. Es handelt sich dabei um folgende Ausgabewerte:

* <E> Mittelwert
* S rms-Wert
* Counts Anzahl der *events*

Diese Größen werden in den Programmen GIL und DAQOUT berechnet (siehe auch Teil II). In diesem Fall kommen die Daten nicht von einem ADC sondern von einem *displaybuffer* [3].
Dann setzen sich die Fehler für Mittelwert und Standardabweichung

* aus den Rundungsfehlern in den Rechenoperationen und
* aus der begrenzten *displaybuffer*-Auflösung zusammen.

Für den Zähler (Counts) hängt die Fehlertoleranz nur von den Rundungsfehlern der Rechenoperationen ab. Es wird wieder mit den Rundungsfehlern in den Rechenoperationen begonnen.

---

3) Es muß angemerkt werden, daß die *Displaybuffer*-Einträge mit den Fehlern auf grund der Kalibration und der ADC-Auflösung behaftet sind. Diese Fehler sind in den folgenden Überlegungen nicht enthalten.

5.2.2.1 Rundungsfehler

Es wurde in Kapitel 5.2.1.1 erwähnt, daß hier praktisch keine Rundungsfehler auftreten.
Wie bei der *single-floating-point*-Darstellung werden ebenfalls

* die Mantisse q,
* der Exponent n und
* das Vorzeichen einer Zahl X gespeichert, wenn

$$|X| = q\ 2^n \text{ , mit } n \in \mathbb{Z} \text{ und } 1 > q \geq 1/2 \text{ ist.}$$

Die Speicherplatzaufteilung ist jedoch eine andere (vgl. Kap. 5.2.1.1)

Vorzeichen	1 Bit
Exponent	8 Bits
Mantisse	55 Bits
	64 Bits = 8 Bytes

Mit einer Mantissengröße von 56 Bits tritt ein Rundungsfehler bei der Berechnung der Standardabweichung (vergleiche oben) entsprechend später auf:

$$2^{56} = \sum_{i=1}^{N} 2048^2 \quad <=> \quad N = 1.7 * 10^{10} .$$

Das heißt, daß frühestens ab $1.7 * 10^{10}$ *events* bei der Berechnung der Standardabweichung Rundungsfehler auftreten können.

Bei der Berechnung des Mittelwertes tritt der Fehler sogar erst ab $N = 3.5 * 10^{13}$ auf.
Da in der Praxis die Anzahl der *events* pro *run* weit unterhalb von 10^{10} liegen wird, werden die obengenannten Ausgabewerte frei von Rundungsfehlern sein.

5.2.2.2 Die *Displaybuffer*-Auflösung

Die begrenzte *displaybuffer*-Auflösung bewirkt einen systematischen Fehler von maximal einem halben *displaybuffer*-Kanal, weil die einzutragenen Werte auf- bzw. abgerundet werden müssen. Die Abbildungen 5.2 a ... c zeigen den unterschiedlichen Einfluß des Kalibrationsfaktors a.

$$\sigma_Y \leq \Delta x \text{ , wobei } 0 \leq \Delta x \leq 1/2 \text{ ist.}$$

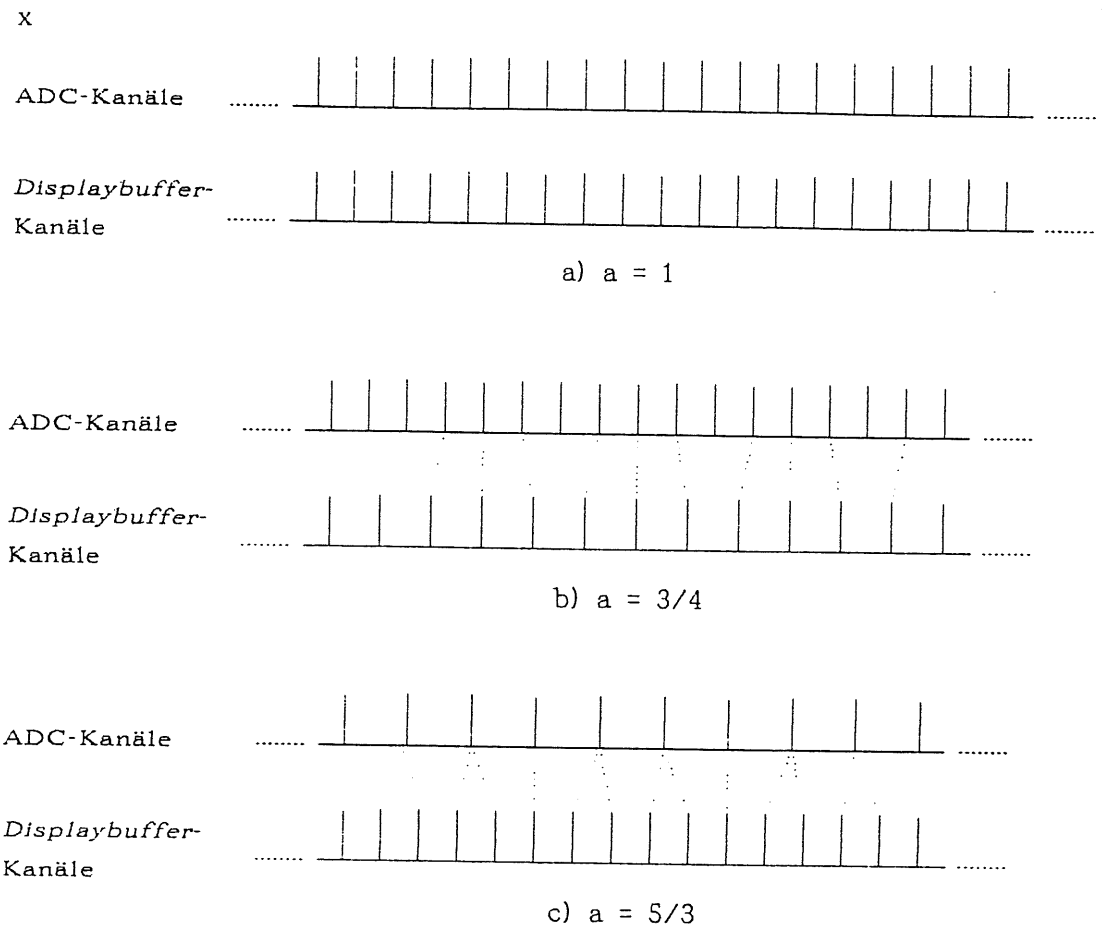


Abbildung 5.2: Rundungsfehler bei der Kanalzuordnung.

Durch diese Abbildungsfunktion, die jeweils einem *displaybuffer*-Kanal eine unterschiedliche Anzahl von ADC-Kanälen zuordnen kann, können die *displaybuffer*-Spektren eine besondere Struktur bekommen (vergleiche z.B. Abb. 7.6 a).

5.2.3 Zusammenfassung der Fehler

In der folgenden Tabelle werden die systematischen Fehler, wie sie sich insgesamt für die verschiedenen Programme aus den vorhergehenden Überlegungen ergeben, dargestellt. Dabei werden Fehler, wenn sie aus mehreren Komponenten zusammengesetzt sind, nach dem Fehlerfortpflanzungsgesetz berechnet. Die Rundungsfehler durch Rechenoperationen werden hier nicht berücksichtigt.

	Systematischer Gesamtfehler $\sigma_Y \leq$	
DAQIN	$\sqrt{\frac{5}{4}}\ a$	
CALIN	1	
CALOUT	1	
GIL	$\sqrt{\frac{6}{4}}\ a$	
DAQOUT Kommandos	$x \leq x_{Thres}$	$x \geq x_{Thres}$
"Dnn", "DS"	$3\,c\,x_0^2 + 2\,d\,x_0 + e$	a
"G","I","M"	$\sqrt{\{3\,c\,x_0^2 + 2\,d\,x_0 + e\}^2 + \frac{1}{4}}$	$\sqrt{\frac{5}{4}}\ a$

Tabelle 5.1: Der systematische Gesamtfehler in verschiedenen Programmen. Alle Angaben sind in *displaybuffer*-Kanaleinheiten, wenn die *displaybuffer*-Auflösung 2048 Kanäle beträgt. Für DAQOUT wurde die Rundungsfehler-Komponente vernachlässigt.

Kapitel 6

Totzeitmessung des Datenaufnahmesystems

Ein wichtiger Gesichtspunkt von Datenaufnahmesystemen ist generell das Verhältnis

$$f' / f \qquad [\, f' \,] = [\, f \,] = \text{Hz},$$

wobei f' die Anzahl der registrierten Impulse pro Zeiteinheit, und
f die Anzahl der eingetretenden Impulse pro Zeiteinheit ist.
Es muß hier angemerkt werden, daß die Zeiteinheit so klein gewählt werden muß, so daß innerhalb eines Zeitintervalls die Impulse kontinuierlich eintreffen. Wenn also in einem Zeitintervall die Impulse teils mit der Rate f_1 und teils mit der Rate f_2 eintreffen, darf hieraus kein mittelres f berechnet werden. Solche Verhältnisse treten in Speicherringen auf, in denen die Teilchen in "Paketen" (*bunch*) gebündelt sind.

Es gilt $f' / f \leq 1$.
Die Verlustrate (f-f') erklärt sich durch die sogenannte Totzeit τ des Aufnahmesystems. Sie entspricht näherungsweise der Verarbeitungszeit des Systems für einen einzelnen Impuls.
In diesem Kapitel werden die Ergebnisse aus Totzeitmessungen des DAQ-Systems (DAQIN[1]) bei verschiedenen Parametereinstellungen vorgestellt. Das Kapitel ist in folgende Abschnitte gegliedert:

- 6.1 Theoretische Berechnung der Totzeitfunktion.
- 6.2 Meßmethoden zur Erfassung von Zählverlusten.
- 6.3 Messungen von τ bei verschiedenen Parametereinstellungen mit periodischen Impulsen. Aus diesen Ergebnissen wird eine Funktion für τ ermittelt, mit der f'/f berechnet werden kann.
- 6.4 Messung des f' mit statistisch verteilten Impulsen. Diese Werte werden den berechneten Zahlen graphisch gegenübergestellt.

---

1) Es wird nur das Datenaufnahmeprogramm DAQIN untersucht. Für die Kalibrations-Software CALIN wird am Schluß des Kapitels eine Abschätzung vorgenommen. Alle anderen Programme haben keine zeitkritischen Teile und werden deshalb auch nicht weiter in Betracht gezogen.

6.1 Theoretische Berechnung der Totzeitfunktion.

Für den Fall, daß die Impulse statistisch verteilt eintreffen, d.h., daß die zeitlichen Abstände der Impulse der Poisson-Statistik genügen, unterscheidet Ruark [RUA37] zwei Typen von Zählern[2]. Eine prägnante Formulierung in deutscher Sprache liefert Jost [JOST46]:
"*Zähler 1. Art:* Massgebend dafür, ob ein Ereignis gezählt wird oder nicht, ist die Zeit, die seit dem letzten *gezählten Ereignis* verflossen ist.
Zähler 2. Art: Massgebend dafür, ob ein Ereignis gezählt wird oder nicht, ist die Zeit, die seit dem letzten *Ereignis* verflossen ist."
Leider hat Jost die Nummerierung von Ruark vertauscht, trotzdem wird im weiteren die Bezeichnung von Jost übernommen.
Danach ist das DAQ-System ein Zähler 1. Art. Die Funktion $f'(f)$ lautet hierfür

$$f'(f) = \frac{f}{1 + f * \tau} \qquad \text{Gl. (6.1)}$$

Dabei ist, wie eingangs schon erwähnt wurde, f' die Rate der registrierten Impulse, und f die Rate der eintretenden Impulse.
Beweis nach Ruark:
Die durchschnittliche Anzahl von Impulsen, die innerhalb der Totzeit τ liegen, ist $f*\tau$. f' entspricht der Anzahl der Zeitintervalle τ pro Sekunde. Daraus folgt, daß $f*\tau*f'$ gerade die Anzahl der verlorenen Impulse pro Sekunde ist.
Dann gilt:

$$f = f' + f\tau f'$$

woraus dann Gl. (6.1) folgt. qed.

Für periodische Impulse ist die Korrekturformel ähnlich der Gl. (6.1):

$$f'(f) = \frac{f}{1 + n(f,\tau)} \qquad \text{Gl. (6.2)}$$

Wobei n die größte ganze Zahl ist, die kleiner oder gleich $f*\tau$ ist.
Gl. (6.2) wird anhand von Abb. 6.1 a,b deutlich.

2) In der Literatur aus den 30er Jahren trat das Problem der Totzeit mehr bei mechanischen und elektronischen Zählwerken und verständlicherweise nicht bei komplexen Datenerfassungssystemen auf. Jedoch ist die Theorie ohne Einschränkung der Allgemeinheit auf die "Zähler" jüngeren Datums übertragbar, weil das zugrunde liegende Modell beide Maschinengenerationen gleichermaßen beschreibt.

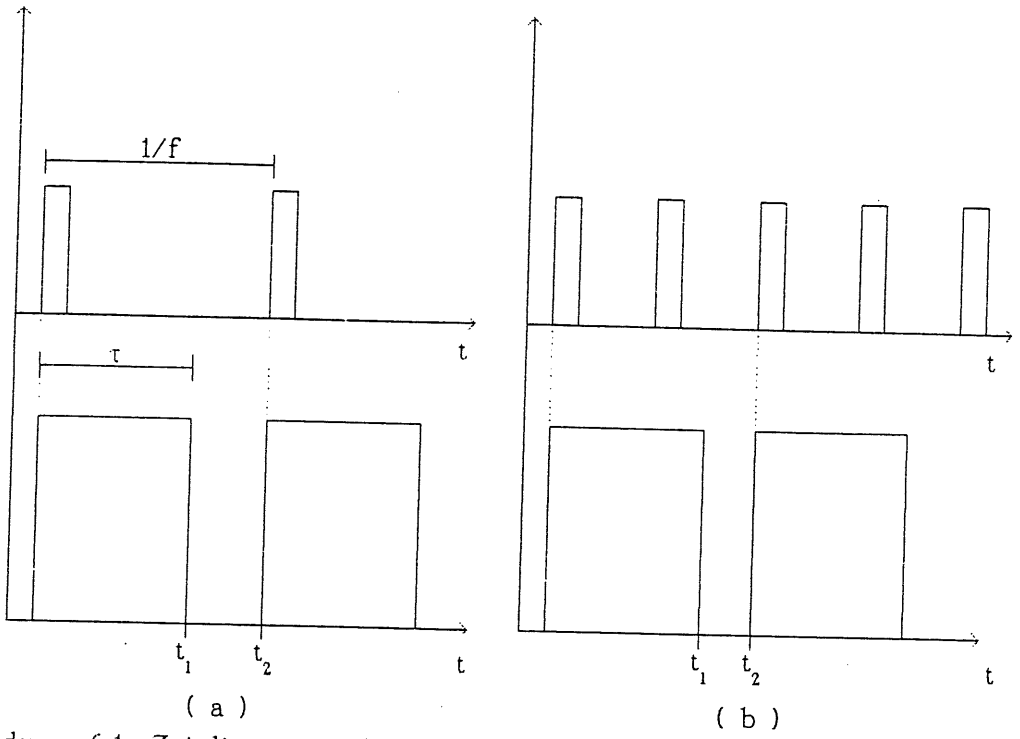


Abbildung 6.1: Zeitdiagramm der Impulse und der ausgelösten Totzeit.
Wenn $\tau < 1/f$ ist, wird jeder Impuls registriert (a). Wenn nun f wächst, so daß $1/f < \tau < 2/f$ wird, so wird nur noch jeder zweite Impuls registriert (b), usw.
Die Zeit zwischen t_1 und t_2 wird als "Wartezeit" bezeichnet.

Aus Gl. (6.1) folgt auch

$$f' = \lim_{f \to \infty} \frac{f}{1 + f * \tau} = \frac{1}{\tau} .$$

Entsprechendes ist auch für Gl. (6.2) gültig.

Der Zähler 2. Art wird hier nicht erörtert. Der Vollständigkeit halber soll dennoch die Korrekturformel angegeben werden:

$$f' = f * e^{-f\tau} \qquad \text{Gl. (6.3)}$$

6.2 Meßmethoden zur Erfassung von Zählverlusten

Es gibt verschiedene bekannte Methoden, die Größe τ einer Zählanordnung zu bestimmen, wie z.B.

* durch Abschwächung der Strahlung einer Quelle [LON48],
* durch Verwendung einer Quelle mit geeigneter Halbwertszeit [FLA39],
* durch die Zweiquellen-Methode [LIF38],
* durch Messung mit zwei Zählrohren, die in definierter Geometrie zu einer Strahlungsquelle angeordnet sind [PET62].

Diese genannten Methoden dienen alle dem gemeinsamen Zweck, in erster Linie die Totzeit eines **Detektors** zu messen. Deshalb arbeiten sie alle mit einer radioaktiven Quelle. Was hingegen die aufgeführten Methoden unterscheidet, ist die Verfahrensweise zur Bestimmung der durchschnittlichen Emissionsrate der Quelle. Die Totzeit der registrierenden Elektronik steht dabei im Hintergrund.

In dem Fall des DAQ-Systems ist der Zweck ein anderer. Hier soll die Totzeit eines registrierenden Systems gemessen werden. Die Erholzeit des Detektors wird zwar berücksichtigt, ist jedoch nicht die zu messende Größe. Das führt zu einer prinzipiellen anderen Methode als die oben erwähnten, da zur Vermessung nun auch "synthetische" Impulse eines Testpulsers mit definierbarer Frequenz verwendet werden können.

Zur Erzeugung von statistisch verteilten Impulsen wird eine α-Quelle Cm_{244} (Die Halbwertszeit beträgt 18.11 a) verwendet, deren Abstand zum Detektor variabel gehalten wird. Dadurch können durchschnittliche Impulsraten zwischen 0-1000 Hz erreicht werden. Die Messung der durchschnittlichen Impulsrate f der Quelle wird mit einem Frequenzmesser ausreichend [3] genau vorgenommen. Abb. 6.2 zeigt die Anordnung für periodische Signale (a) und für poissonververteilte Signale (b).

---

3) Der Fequenzmesser kann Impulsabstände von 1 µs unterscheiden. Die Erholzeit des verwendeten Halbleiterdetektors beträg ca. 10ns [BOR85] und kann deshalb vernachlässigt werden. Die Anzahl der Impulse, die vom Frequenzmesser nicht registriert werden können, wird im folgenden abgeschätzt.
Die Impulse der α-Quelle genügen der Poisson-Statistik. Es wird Gl. (6.1) benutzt, wobei τ = 1 µs und f = 1000 Hz ist. Somit wird

$$f'/f = 0.999 \quad , \text{d.h.,}$$

0.1 % der Impulse werden wegen der Totzeit nicht registriert.
Widerum beträgt eine Messdauer 100s, so daß in dieser Zeit bei 1000 Hz 100 000 Impulse (N) gezählt werden. Die statistische Schwankung des Werts für f' beträgt dann $1/\sqrt{N}$ = 0.3 %. Weil die statistische Schwankung für N dreimal so groß ist wie der Totzeiteffekt, macht es keinen Sinn die obengenannten verlorenen Impulse in einer Korrekturrechnung mit aufnehmen zu wollen,

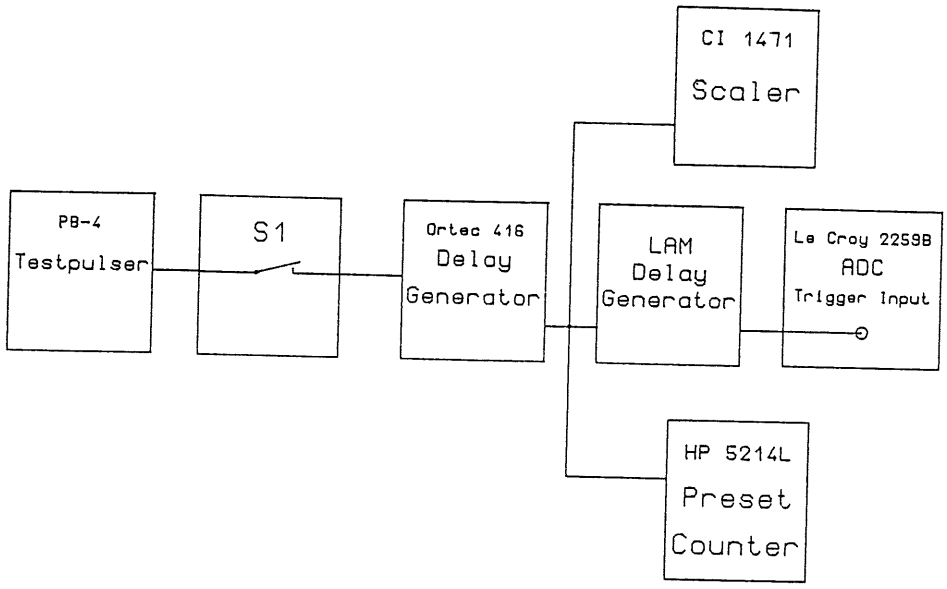


Abbildung 6.2 a: Periodische Impulse

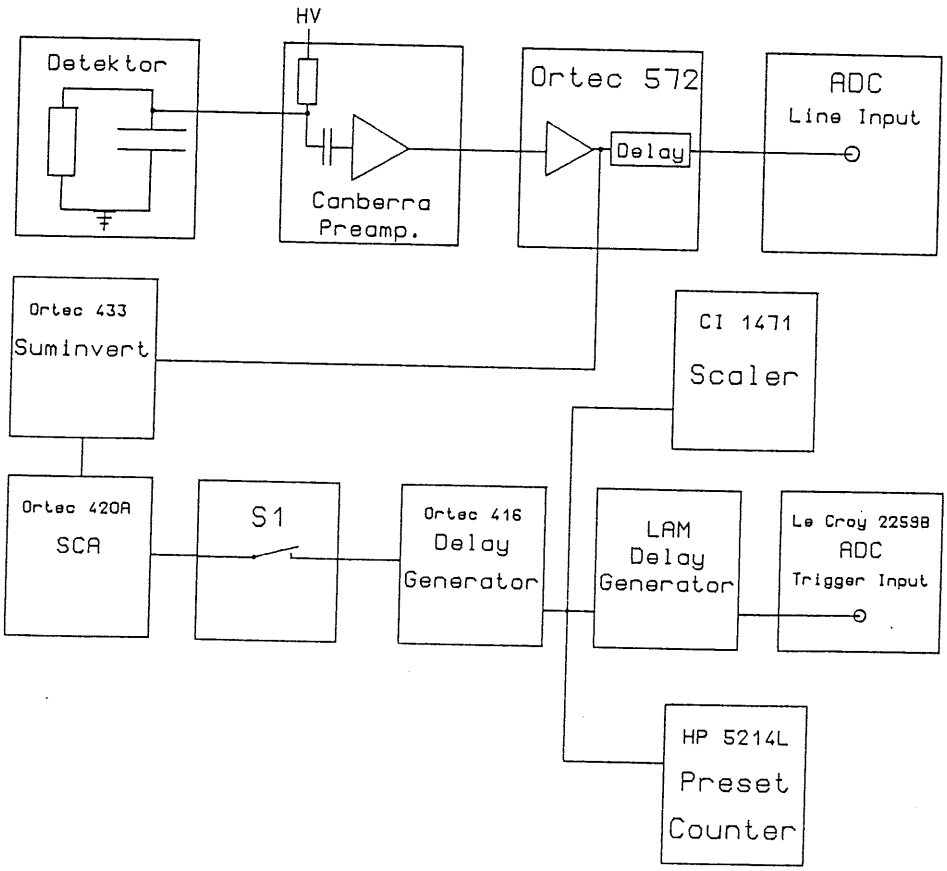


Abbildung 6.2 b: Statistisch verteilte Impulse

Abbildung 6.2: Schematischer Aufbau der elektronischen Instrumente.
a) Periodische Impulse und b) statistisch verteilte Impulse.

Um den Scaler und das DAQ-System synchron starten und stoppen zu können, wird der Schalter S_1 benötigt. Bei geöffneter Schalterstellung werden der Scaler und das Programm DAQIN "scharf" gemacht. Wenn nun S_1 geschlossen wird, werden die ankommenden Impulse mit dem Scaler gezählt und gleichzeitig an das DAQ-System weitergegeben (LAM-Generator) und dort ebenfalls gezählt. Außerdem wird die Frequenz der ankommenden Impulse gemessen (Preset Counter). Öffnet man den Schalter wieder, können die Werte n am Scaler und n' am DAQ-System abgelesen werden. Die Rate f mußte während der Meßdauer am Preset Counter ablesen werden. Daraus folgt

$$f' = f \frac{n'}{n} .$$

6.3 Messung mit periodischen Impulsen

Die Verarbeitungszeit τ für ein *event* ist keine Konstante, sondern eine von mehreren Parametern abhängige Größe. Da die Aufgaben innerhalb des Programms nur von einem Prozessor bewältigt werden, läßt sich die Bearbeitungzeit eines bestimmten Programmabschnitts abschätzen. Der Programmabschnitt, der hier zur Diskussion steht, ist die CAMAC-Routine des Programms DAQIN (Abb. 3.7). Diese Routine muß durchlaufen werden, wenn ein *event* vom LAM-Generator registriert worden ist. Sie läßt sich nun grob in vier Abschnitte einteilen.

* Ein Teil, der N_A-mal durchlaufen werden muß. N_A ist die Anzahl der ADCs, die für die *list-mode*-Abspeicherung konfiguriert worden sind.
* Ein zweiter Teil, der N_D-mal durchlaufen werden muß. N_D ist die Anzahl der ADCs, für die jeweils ein *displaybuffer* erzeugt wird.
* Ein Teil, der nur durchlaufen werden muß, wenn das *lifedisplay* aktiv ist und schließlich
* ein Restteil der in jedem Fall durchlaufen werden muß.

Zur Totzeit kommt noch die Konvertierungszeit τ_K der verwendeten ADCs hinzu. Die notwendige Verzögerung wird durch den LAM-Generator gewährleistet (siehe auch Kap. 3.1).
Somit setzt sich die Verarbeitungszeit τ für ein *event* wie folgt zusammen:

$$\tau = \tau_K + N_A \tau_A + N_D \tau_D + L \tau_L + \tau_R \qquad \text{mit } L = \begin{cases} 1 & \text{für Lifedisplay on} \\ 0 & \text{für Lifedisplay off} \end{cases} \qquad \text{Gl. (6.4)}$$

Zwei Einflüsse auf τ, die weiter unten gesondert behandelt werden, bleiben in der Gl. (6.4) unberücksichtig:

* τ ist abhänig von den ausgelesenen ADC-Werten.
* In regelmäßigen Abständen müssen Daten vom RAM auf die Platte kopiert werden. Für die Dauer der Kopierzeit ist das System ebenfalls blockiert.

Die Gl. (6.4) wird nun sukzessiv diskutiert. Dabei werden die Zeitkomponenten τ_K, τ_A, τ_D, τ_R und τ_L der Gl. (6.4) bestimmt.
Um in den Messungen sicherzustellen, daß nur die CAMAC-Routine des Programms DAQIN und nicht andere Programmteile duchlaufen wird, wurde die Eingangsimpulsfrequenz genügend hoch eingestellt. Dadurch wird gewährleistet, daß bei der Abfrage des LAM-Signals (siehe Abb. 3.5) immer zur CAMAC-Routine verzweigt wird. Das erfüllte gleichzeitig die Forderung, den Impulsabstand und damit die Wartezeit (siehe Abb. 6.1) möglichst gering zu halten.

Es wird mit τ (τ_K) begonnen.
Die restlichen Parameter blieben während der Messung unverändert ($N_A = 1$, $N_D = 1$, *lifedisplay* inaktiv).
Der LAM-Generator ist auf Verzögerungen zwischen 0 ... 255 µs programmierbar und τ müßte linear mit dem eingestellten Wert wachsen.

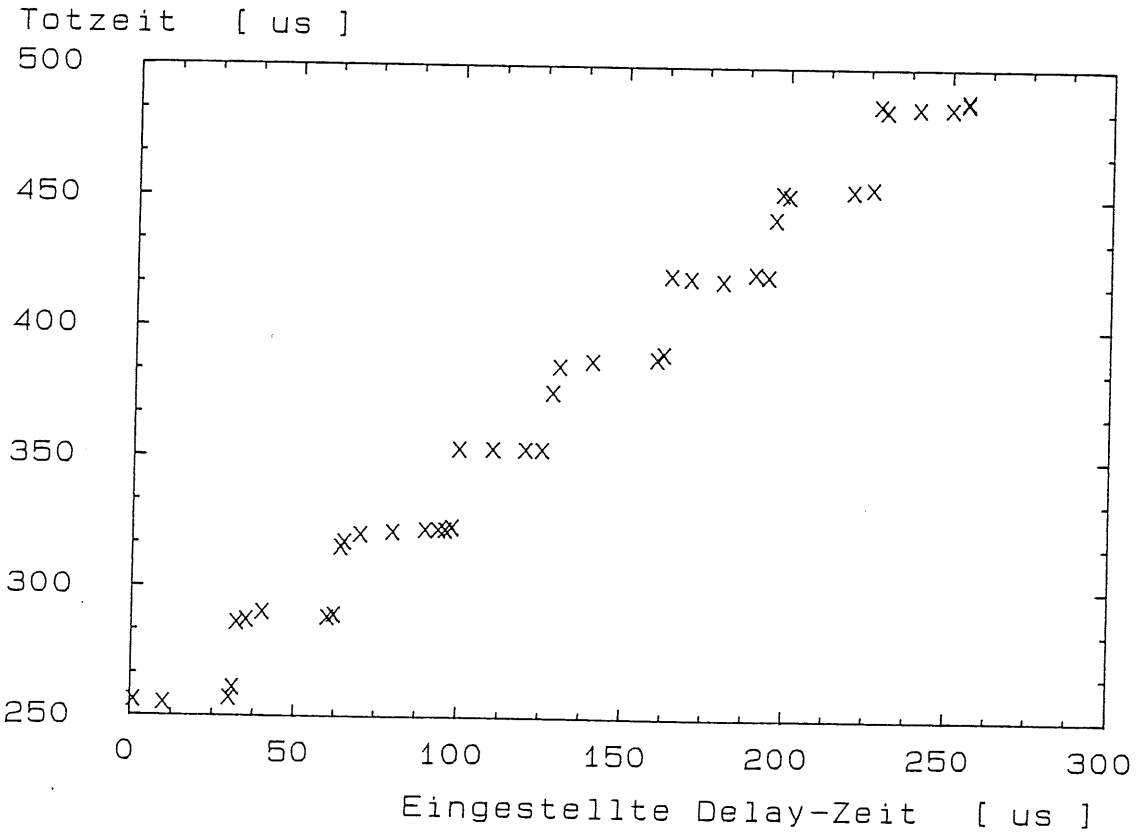


Abbildung 6.3: Abhängigkeit der Totzeit von der Größe τ_K.

In Abb. 6.3 ist zu sehen, daß unerwarteterweise keine Gerade mit der Steigung 1 entsteht, sondern eine Treppenfunktion mit der mittleren Steigung 1 und einem Offset von ca 255 µs. Auf der Suche nach Erklärungen dieses Phänomens wurde die gleiche Messung mit einer stark gekürzten CAMAC-Routine durchgeführt. Der herausgeschnittene Programmteil ist von der eingestellten Verzögerungszeit unabhängig, d.h., es kann prognostiziert werden, daß sich bei einer erneuten Messung der Schnittpunkt der Treppenfunktion mit der Y-Achse verschieben sollte, die Steigung und die Treppenform aber konstant bleiben müßte.
Das Resultat wird in Abb. 6.4 wiedergegeben.

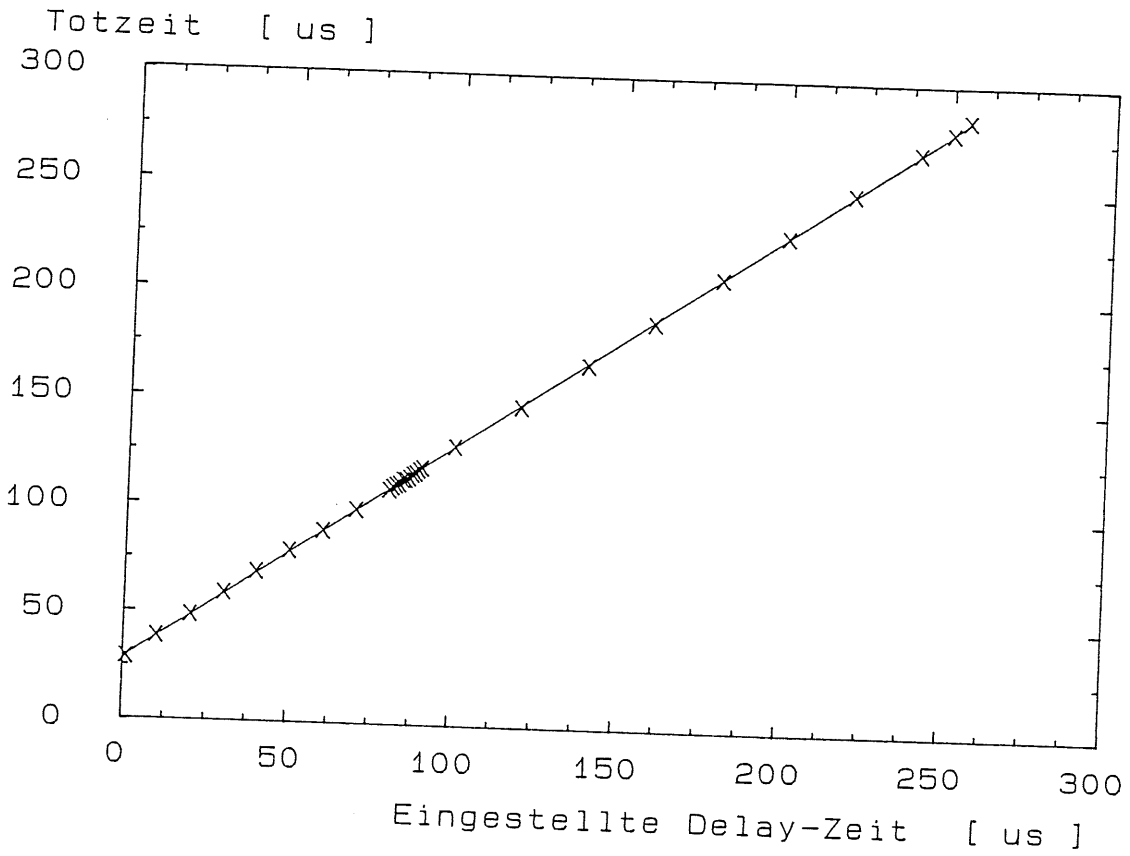


Abbildung 6.4: Abhängigkeit der Totzeit von der Größe τ_K mit reduzierter CAMAC-Routine.

Nun ergibt sich aber die Funktion, die zuerst erwartet wurde. Vermutlich entsteht die Treppenfunktion aus Gründen, die aus der Rechnerarchitektur resultieren. In dem zeitlichen Rahmen dieser Arbeit konnte dieses Phänomen nicht umfassend geklärt werden, es bleibt jedoch festzuhalten, daß die Steigung in beiden Fällen den Erwartungen entsprechend 1 ist.

Bevor der nächste Parameter untersucht werden kann, müssen einige Bemerkungen zu dem Einfluß der ausgelesenen ADC-Werte auf die Totzeit τ gemacht werden. Innerhalb der CAMAC-Routine werden eine Reihe von *floating-point*-Rechenoperationen durchgeführt. Die Ausführungszeit dieser Operationen hängt von den Beträgen der Operanden ab, und die Operanden sind wiederum von den ADC-Werten abhängig. So wird für die Ausführungszeit z.B. einer *floating-point*-Addition minimal 8.3 µs, maximal 31.7 µs und ein durchschnittlicher Wert von 9.3 µs angegeben [KDJ84]. Da der Anteil an *floating-point*-Operationen in der CAMAC-Routine groß ist, würden Meßwerte für τ_A und τ_D sehr stark von den Beträgen der Operanden bzw. der ADC-Werte abhängen. Um für τ_A und τ_D reproduzierbare Meßwerte zu erhalten, wurden Messungen mit und ohne *floating-point*-Operationen in der CAMAC-Routine vorgenommen. Um bei den Messungen mit *floating-point*-Operationen dafür zu sorgen, daß immer die selben Operanden verwendet werden, wurde das ADC aus dem CAMAC-Rahmen gezogen, so daß der offene CAMAC-Bus, d.h. immer der Wert 0, ausgelesen wurde. Eine Ausnahme bildeten die *Counts*, weil sich bei der Addition der *events* natürlich die Operanden ändern. Messungen mit *random inputs* wurden auch durchgeführt und werden im Abschnitt 6.4 behandelt.
Die Messung ohne *floating-point*-Operationen wurde durch Entfernen der entsprechenden Anweisungen in der CAMAC-Routine vollzogen. Dabei bildeten die *Counts* wieder eine Ausnahme: Diese Variable wurde als *floating-point*-Größe beibehalten, da sie ein zu messender Wert war.

Es wird nun die τ -Abhängigkeit von N_A untersucht. Die übrigen Parameter hatten folgende Werte: $N_D = 1$, $\tau_K = 1\,\mu s$, kein *lifedisplay*.

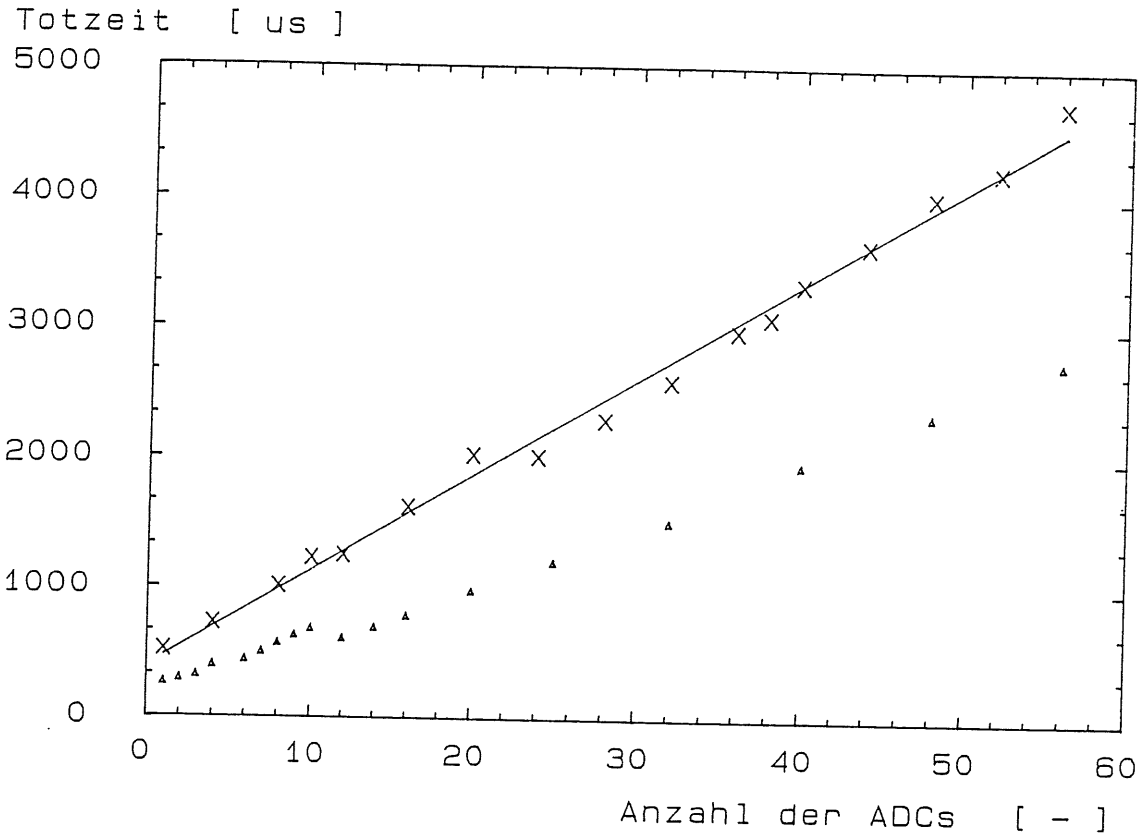


Abbildung 6.5: Abhängigkeit der Totzeit von der Größe N_A.
x : Mit *floating-point*-Operationen.
△ : Ohne *floating-point*-Operationen.

In Abb. 6.5 ist deutlich zu erkennen, daß die *floating-point*-Operationen die Totzeit erheblich verlängern. Abweichungen von der Linearität führen zu den gleichen Vermutungen wie oben bei der Treppenfunktion. Die Regressionsfunktion lautet:

$$\tau = N_A * 73.6\,\mu s + 395\,\mu s \pm 109\,\mu s \qquad \text{Gl. (6.5)}$$

Aus der Steigung der Regressionsgeraden kann τ_A bestimmt werden:

$$\tau_A = 73.6\,\mu s.$$

Als nächstes wird $\tau(N_D)$ untersucht.
Die übrigen Parameter waren $N_A = 1$, $\tau_K = 1\ \mu s$ und *lifedisplay* inaktiv.

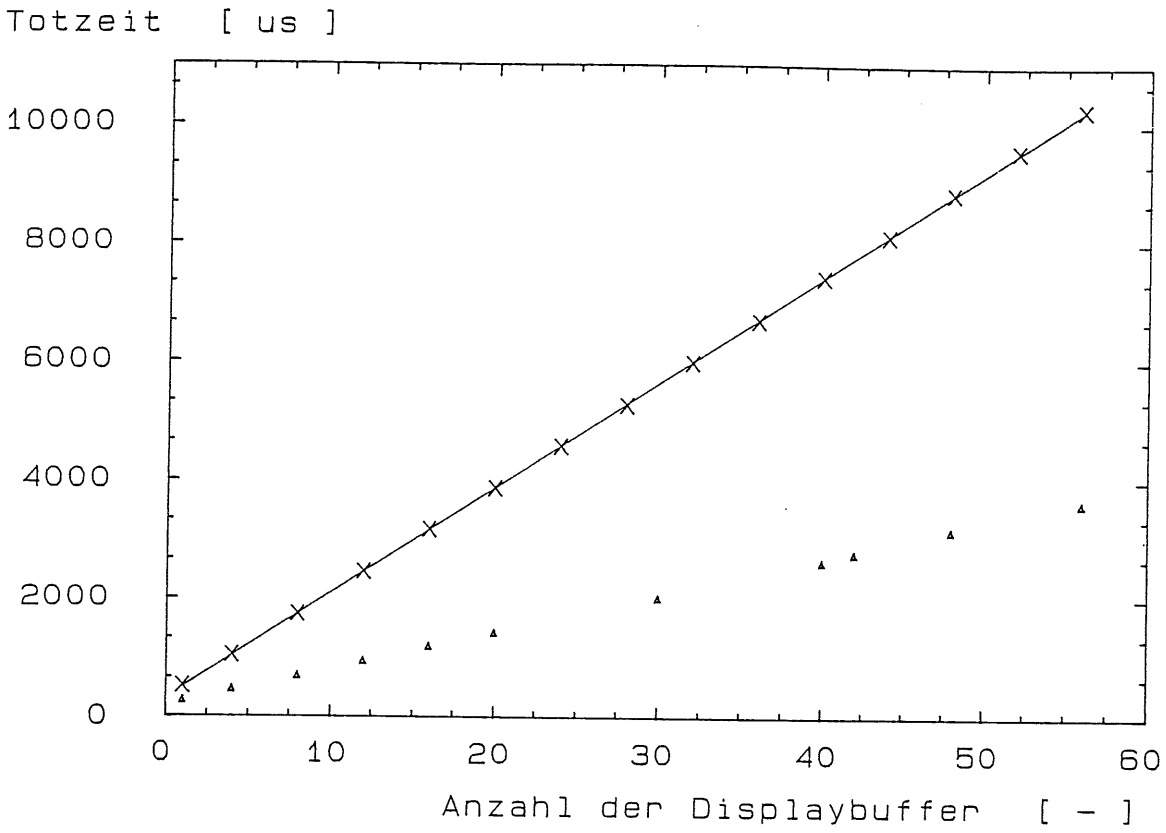


Abbildung 6.6: Abhängigkeit der Totzeit von der Größe N_D.
x : Mit *floating-point*-Operationen.
△ : Ohne *floating-point*-Operationen.

Hier lautet die Regressionsfunktion:

$$\tau = N_D * 177\ \mu s + 317\ \mu s \pm 15\ \mu s \qquad \text{Gl. (6.5)}$$

Aus der Steigung der Regressionsgeraden in der Abbildung 6.6 läßt sich das τ_D bestimmen.

$$\tau_D = 177\ \mu s .$$

Im Vergleich zu τ_A wird in diesem Programmteil wesentlich mehr Zeit für ein Schleifendurchlauf benötigt. Ursache ist dafür die ständige Aktualisierung von Mittelwert und Standardabweichung, die in dem anderen Programmteil nicht vorgenommen werden muß.

Jetzt wird τ_R aus der Gl. (6.6) und Gl. (6.4) bestimmt.

$$\tau_R = 242\ \mu s$$

An dieser Stelle soll die Wirkung des Kopierens vom RAM auf die Platte erläutert werden, wei sie den Wert τ_R mitbeeinflußt. τ_R ist ein Durchschnittswert, deshalb gilt Gl. (6.2) nur für $f \gg f'$. Für $f \approx f'$ wird der Meßwert f'/f schlechter ausfallen als Gl. (6.2) voraussagt, was an einem Beispiel und anhand der Abb. 6.7 deutlich gemacht werden soll:
Es sei angenommen, daß $\tau < 1/f$ sei. Danach müßte nach Gl. (6.4) $f' = f$ sein. Nun ist aber τ_R ein Mittelwert aus τ_R' für **n** *events* und $\tau_R' + \tau_C$ für **ein** *event*. τ_C sei hierbei die Kopierzeit der *list-mode*-Daten vom RAM auf die Festplatte. Da nun im allgemeinen $\tau_C > 1/f$ ist ($\tau_C \approx 36$ ms), gehen während dieser Kopierzeit mit Sicherheit mehrere *events* verloren.

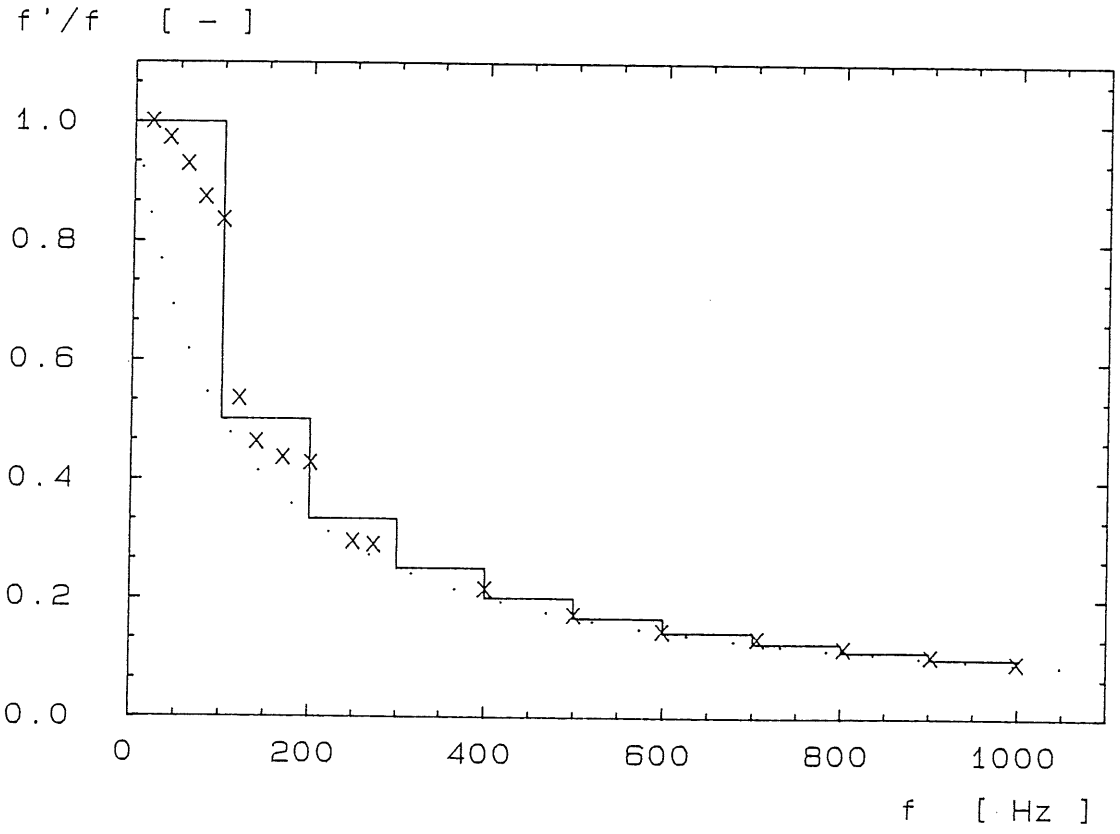


Abbildung 6.7: f'/f mit periodischen Signalen.
Konstante Parameter: $N_A = 56$, $N_D = 32$, $\tau_K = 1\ \mu s$, *lifedisplay* inaktiv.
Die Kreuze sind Meßwerte. Die durchgezogene Kurve ergibt sich aus Gl. (6.2) mit $\tau = 10$ ms. Die punktierte Kurve ergibt sich aus Gl. (6.1) ebenfalls mit $\tau = 10$ ms.

Zuletzt wird τ_L bestimmt.
Die anderen Parameter wurden wie folgt gewählt:
$N_A = 1$, $N_D = 1$ und $\tau_K = 255$ µs. Es wurden Messungen mit und ohne aktivem *lifedisplay* durchgeführt. Es resultiert:

$$\tau_L = 5157 \text{ µs} \pm 76 \text{ µs}.$$

τ_L erklärt sich vornehmlich aus der seriellen Datenverbindung zwischen dem Rechner und dem Terminal. Für den Eintrag eines Punktes in ein Spektrum müssen 11 Zeichen zum Terminal übertragen werden. Für ein Zeichen werden 9 Bits übertragen. Bei einer Übertragungsrate von 19200 Bits/s würde man für $\tau_L \geq 5156$ µs erwarten.

Zusammenfassend wird Gl. 6.4 für L = 0

$$\tau = (73.6 \text{ µs})\, N_A + (177 \text{ µs})\, N_D + \tau_K + 220 \text{ µs} \pm 110 \text{ µs}$$

und für L = 1

$$\tau = (73.6 \text{ µs})\, N_A + (177 \text{ µs})\, N_D + \tau_K + 5157 \text{ µs} + 220 \text{ µs} \pm 134 \text{ µs}$$

Die hier angegebenen Fehler wurden nach dem Fehlerfortpflanzungsgesetz aus den Einzelfehlern für τ_L und den Gleichungen (6.5) und (6.6) bestimmt.

Messungen aus dem April 1988, als das System seinen Endzustand noch nicht erreicht hatte, ergaben

$$\tau = (79.6 \text{ µs})\, N_A + (201 \text{ µs})\, N_D + \tau_K + (5423 \text{ µs})\, L + 242 \text{ µs} \pm 5\% .$$

Die Meßmethode im April 1988 war ein andere, so daß der Fehler hier mit 5% angegeben werden muß. Die Totzeitfunktion aus den April-Messungen wird in Abb. 6.8 a-b dargestellt. Dabei wurde unter Verwendung der Gl. (6.2) f'/f bestimmt und daraus die relative Totzeit (1 - f'/f) ermittelt. Man erkennt, daß bei Überschreiten gewisser Grenzen nur noch jedes 2., jedes 3. usw. *event* registriert wird.

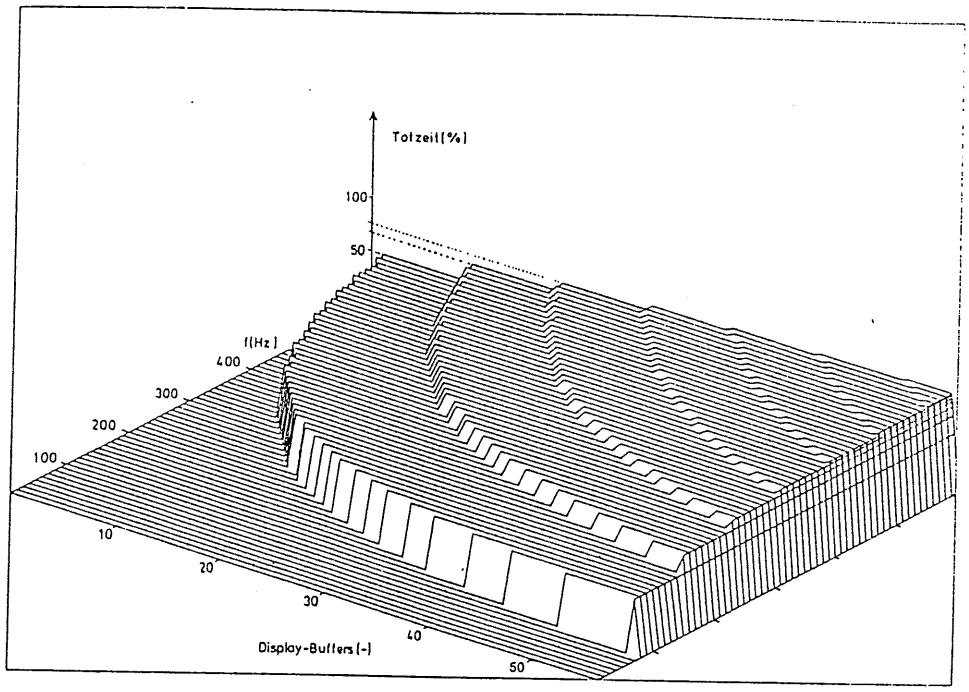


Abbildung 6.8a: Relative Totzeit als Funktion von f und N_D. Die übrigen Parameter waren: N_A = 15, τ_K = 1 µs und *lifedisplay* inaktiv.

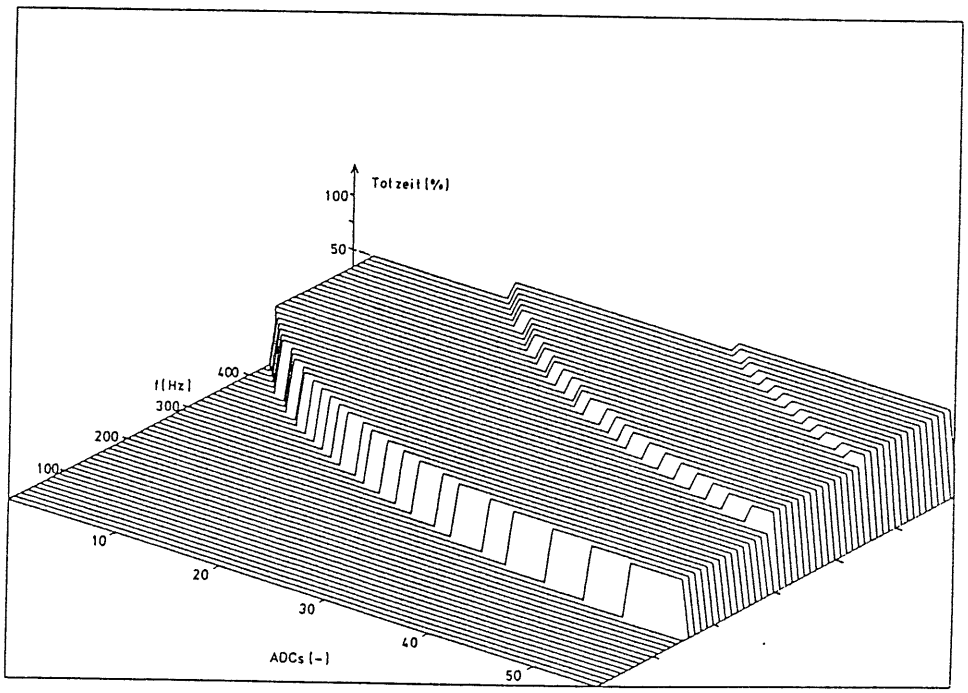


Abbildung 6.8b: Relative Totzeit als Funktion von f und N_A. Die übrigen Parameter waren: N_D = 10, τ_K = 1 µs und *lifedisplay* inaktiv.

6.4 Messungen mit statistisch verteilten Impulsen

In diesem Abschnitt des Kapitels werden

* erstens, Meßergebnisse, die mit gezogenem ADC (siehe oben) erreicht werden, vorgestellt.
* zweitens, werden Ergebnisse aus Messungen mit *random inputs* (in der Höhe variirende Signale) gezeigt.
* drittens, werden in Hinblick auf die spätere Anwendung des DAQ-Systems Ergebnisse mit der schnellsten und mit der langsamsten Konfiguration vorgelegt.

Alle Meßdaten werden mit errechneten Werten verglichen.

Punkt 1:
Wegen der maximalen Emissionsrate der α-Quelle von ≈ 1000 Hz wurde eine Parameterwahl getroffen, bei der sich τ = 10 ms ergab. Die Parameter im einzelnen:
N_A = 56, N_D = 32, τ_K = 1 µs, *lifedisplay* inaktiv.
In Abb. 6.9 werden theoretische und experimentelle Daten einander gegenübergestellt.

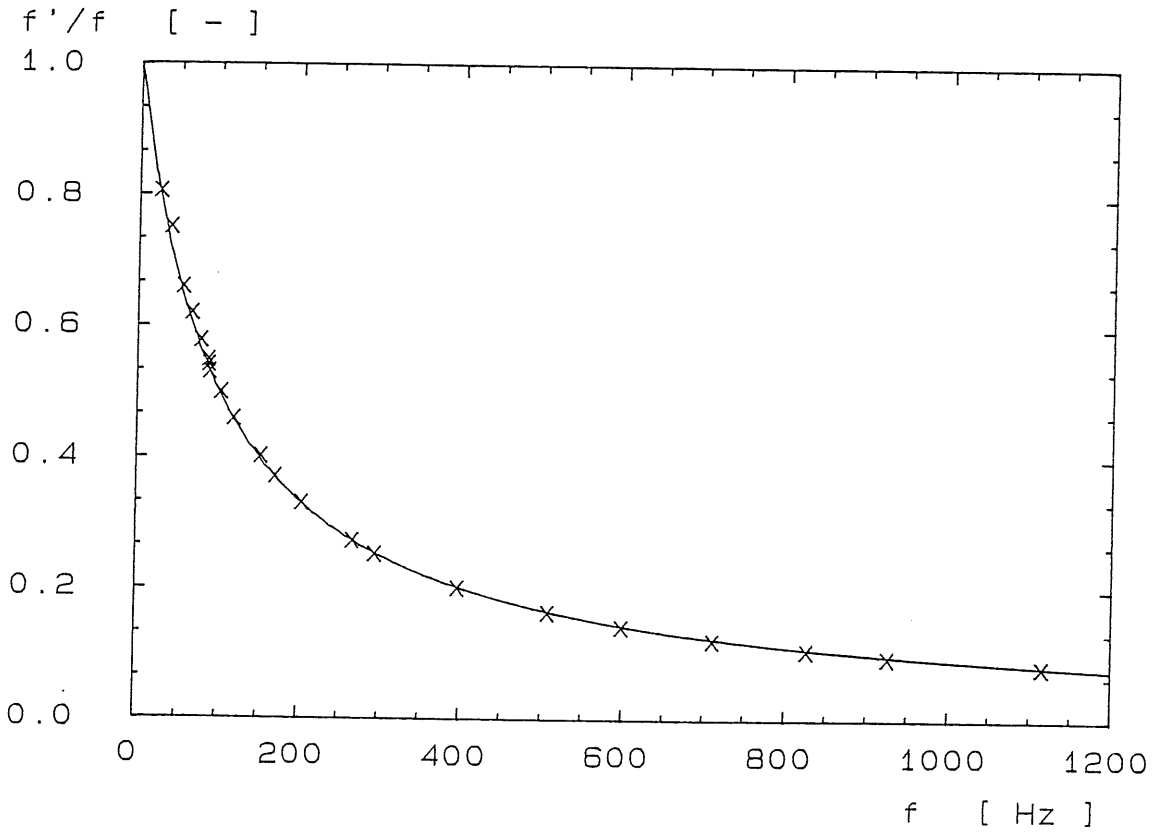


Abbildung 6.9: f'/f als Funktion von f mit τ = 10 ms nach Gl. (6.1).
Konstante Parameter waren: N_A = 56, N_D = 32, τ_K = 1 µs, *lifedisplay* inaktiv.

Punkt 2:

Diese Messung wurde ein zweites Mal mit *random inputs* durchgeführt. Dazu wurde der ADC in den CAMAC-Rahmen gesteckt. *Random inputs* bedeutet in diesem Zusammenhang, daß maximal 2048 verschiedene ADC-Werte ausgelesen wurden. Natürlich konnten damit nicht alle Operandenkombinationen verwirklicht werden. Bei einer 32-Bit Darstellung eines Realwertes können 2^{32} verschiedene Werte für **einen** Operanden dargestellt werden.

Ein ADC-Wert werde mit x bezeichnet. Die Operanden der nach Auslesen eines ADCs folgenden Rechenoperationen nehmen dann ständig wechselnde Werte an, weil jeweils ein Operand den Wert $\sum x_i$ oder $\sum x_i^2$ beinhaltet. Aus diesem Grund darf man von *random inputs* sprechen.

Im vorigen Abschnitt 6.3 wurden Schwankungen für τ_A und τ_D für *random inputs* angegeben (27.8 µs bzw. 120 µs). Mit den gewählten Parameter ergibt sich daraus für τ folgender theoretische Wert:

$$\tau = 10 \pm 5.3 \text{ ms} .$$

Das Vorzeichen der Schwankung hängt davon ab, ob der Wert 0 für die Operanden eine kurze oder lange Rechenzeit für eine *floating-point*-Operation verursacht.

Aus der Abb. 6.10. kann entnommen werden,

* erstens, daß der Wert 0 als Operand kurze Rechenzeiten verursacht, und
* zweitens, daß die Totzeit um 16% gegenüber der ersten Messung gestiegen ist: $\tau_{random} = 11.6$ ms .

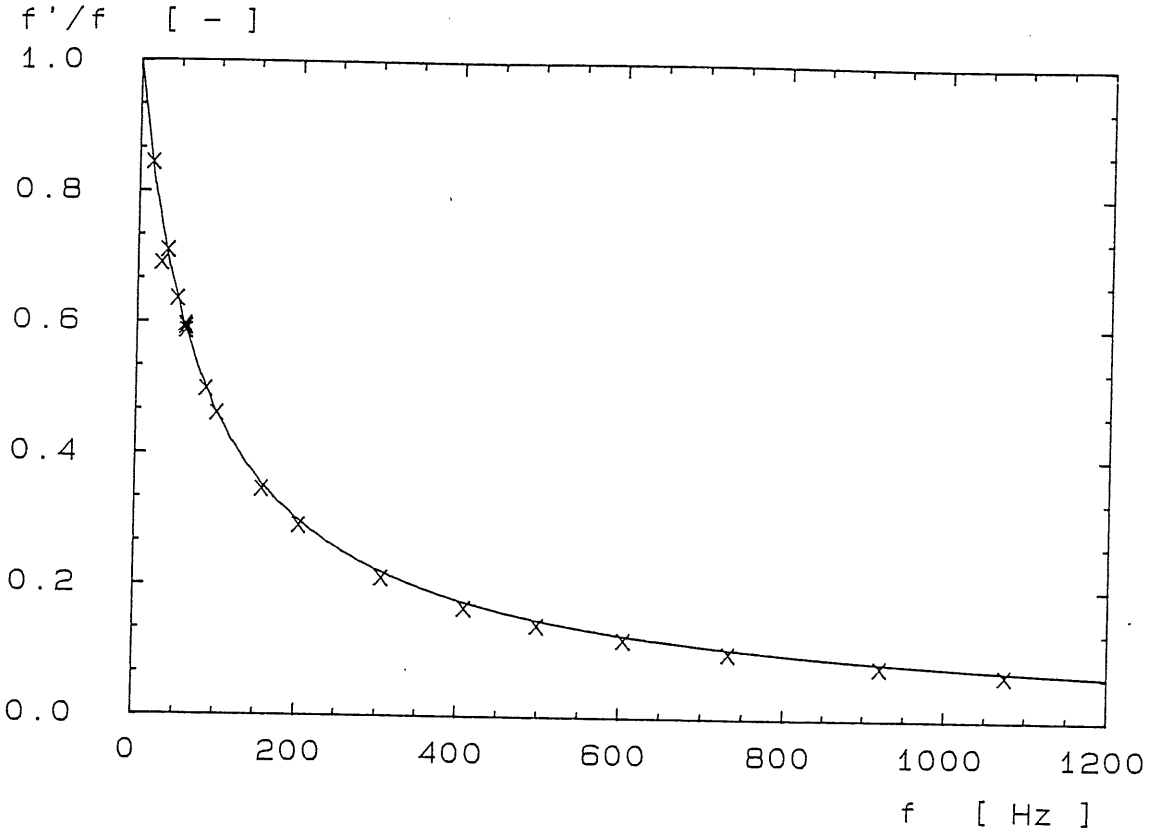


Abbildung 6.10: f'/f als Funktion von f mit τ = 11.6 ms und *random inputs*. Konstante Parameter: N_A = 56, N_D = 32, τ_K = 1 µs, *lifedisplay* inaktiv.

Am Schluß dieses Kapitels werden mit Blick auf die spätere Anwendung der minimale und der maximale Wert für τ angegeben. Mit den Parametern N_A = 1 und N_D = 1 wird der minimale Wert für τ erreicht. Für τ_K werden 255 µs gewählt, was der Verwendung eines ADCs mit langer Konvertierungszeit entspricht. Berechnet aus Gl. (6.4) ergibt sich für τ:

$$\tau_{min} = 748\ \mu s\ .$$

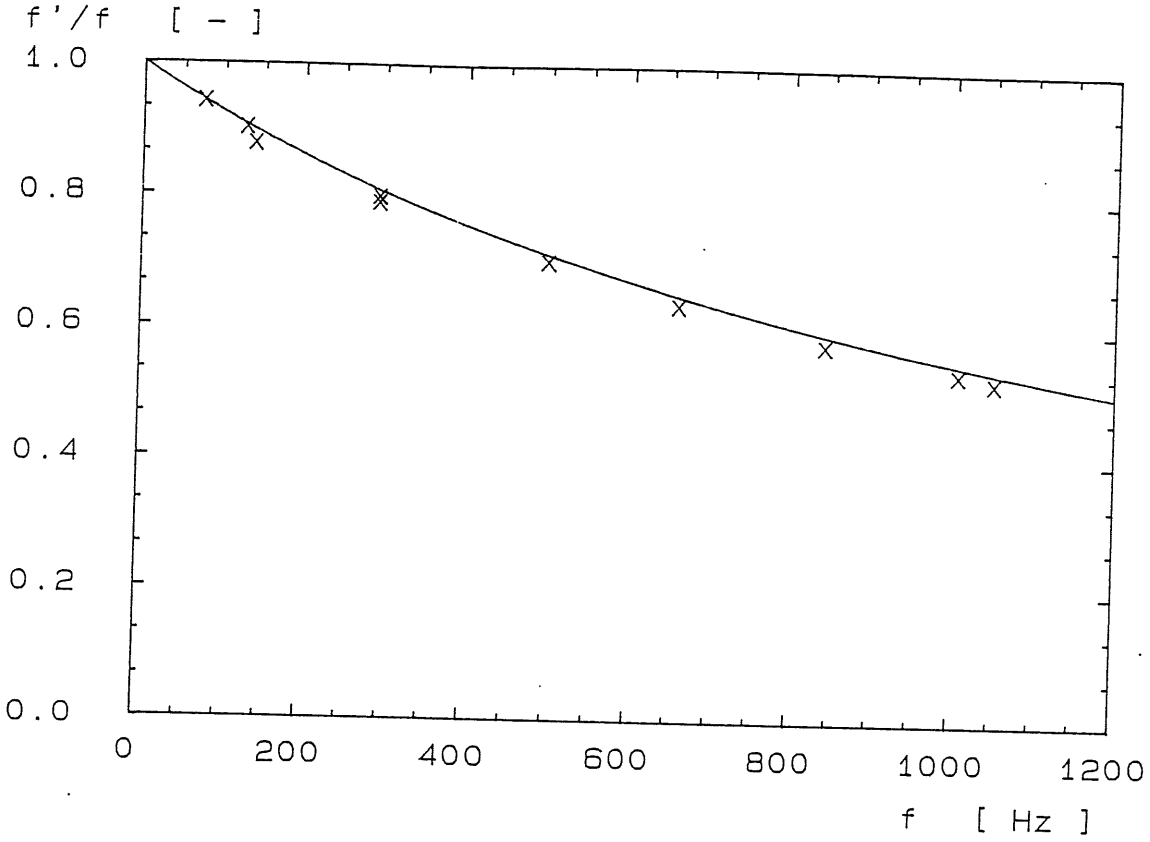


Abbildung 6.11: f'/f als Funktion von f mit τ_{Random} = 814 µs .
Konstante Parameter waren: N_A = 1, N_D = 1, τ_K = 255 µs, *lifedisplay* inaktiv.

Abb. 6.12 gibt das Verhalten für die rechenaufwendigste Konfiguration mit N_A = 56, N_D = 56, τ_K = 255 µs und inaktivem *lifedisplay* wieder. Für τ folgt aus Gl. (6.4):

$$\tau_{max} = 14.5 * 10^3 \text{ µs} .$$

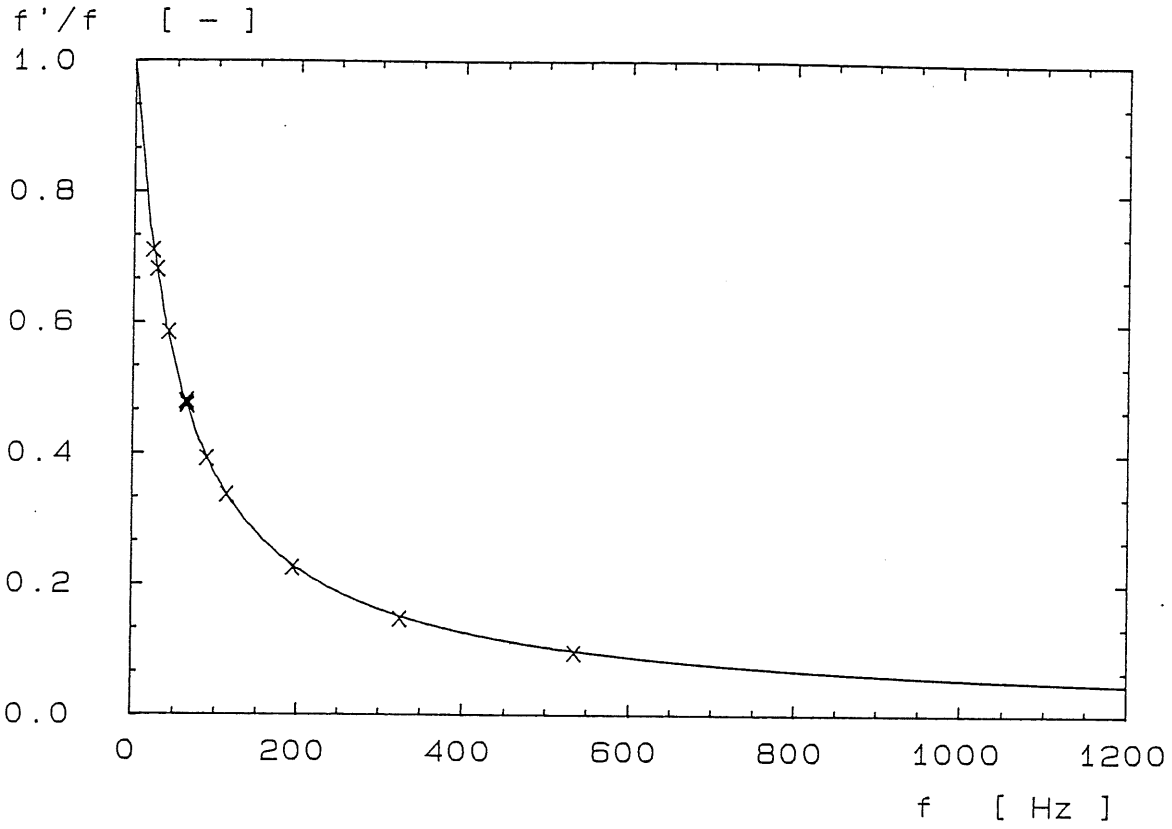


Abbildung 6.12: f'/f als Funktion von f mit τ_{Random} = 17.4 ms .
Konstante Parameter sind: N_A = 56, N_D = 56 , τ_K = 255 µs, *lifedisplay* inaktiv.

Die Kalibrations-Software CALIN wurde nicht vermessen, da es für Testpulser und für eine Impulsfrequenz von 25 Hz vorgesehen ist, und weil für diese Frequenz die Quotient f'/f ≥ 0.98 ist, denn:
Wenn die CAMAC-Routinen von DAQIN und CALIN miteinander verglichen werden (siehe Abb. 3.7 und Abb. 3.6), so muß die Totzeit von CALIN (τ_{CAL}) vergleichbar sein mit der Totzeit von DAQIN (τ) mit den Parametern N_A = 1 und N_D = 1. Tatsächlich muß sie etwas günstiger ausfallen, weil es kein Summenspektrum zu verwalten gibt. Es kann daher τ_{CAL} abgeschätzt werden zu

$$\tau_{CAL} \leq 814\ \mu s.$$

Kapitel 7

Testexperiment am Isochronzyklotron

In diesem Kapitel wird der Einsatz des DAQ-Systems vor Ort in einem Experiment beschrieben. Dieser Einsatz erfüllte eine Doppelfunktion.

* Erstens sollte die Anpassung der Prozessabläufe an den experimentellen Erfordernissen untersucht und ggf. optimiert werden.
* Zweitens mußte die Messung im Rahmen einer anderen Diplomarbeit durchgeführt werden.

Bei diesem Experiment wurden 8 Siliziumdetektoren hintereinander zu einem Kalorimeter angeordnet. Als nachzuweisende Teilchen wurden Protonen mit einer Energie von 22.8 MeV gewählt. Die Energie wurde bewußt so gewählt, daß das Silizium aller 8 Detektoren ausreicht, um die Protonen zu stoppen. Auf diese Weise kann auf Absorbermaterial verzichtet werden und man erhält dadurch ein ideales Kalorimeter, das nur aus aktivem Material besteht. Mit diesem Kalorimeter kann die Einschußenergie und die Strahlbreite nachgewiesen werden, weshalb auch geringe Fehler in der gesamten Meßapperatur einschließlich des DAQ-Systems vermessen werden können.
Die vielen Erkenntnisse[1)], die zu einer Modifikation des Systems führten, werden in diesem Kapitel nicht im einzelnen aufgezeigt, sondern es sollen vielmehr am Beispiel einer Einzelmessung im Rahmen des Experiments die für das DAQ-System ableitbaren Ergebnisse zusammengefaßt werden.

1) Eine besondere Problematik führte sogar zu einer Erweiterung des Systems und ist in Kapitel 4 näher beschrieben.

7.1 Versuchsanordung

Acht Oberflächen-Sperrschicht-Detektoren wurden hintereinander in einem *stack* zu einem Kalorimeter zusammengefaßt und mit 22.8 MeV Protonen beschossen. Absorbermaterial zwischen den Detektoren wurde nicht verwendet.
In Abbildung 7.1 wird der schematische Strahlenverlauf gezeigt.

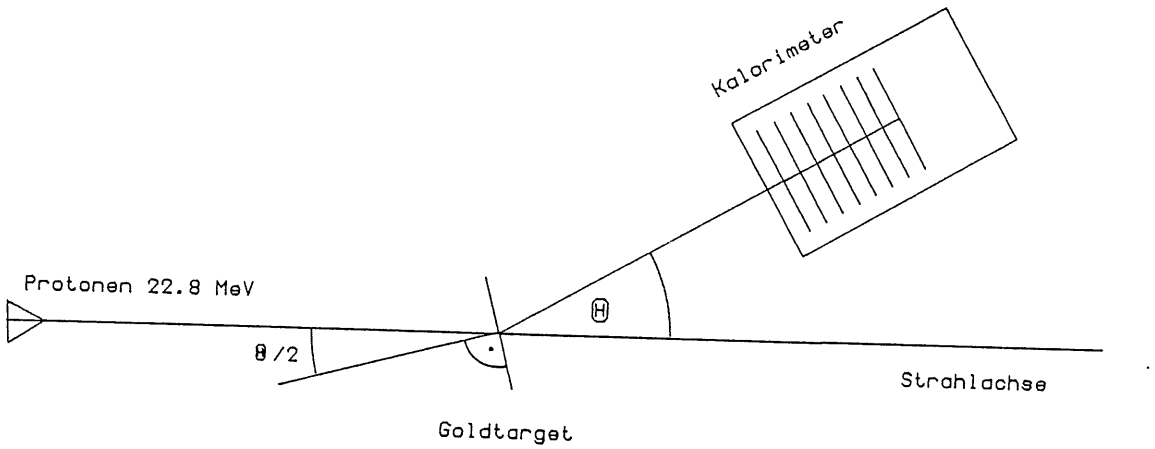


Abbildung 7.1: Schematischer Verlauf des Strahles.

Der Primärstrahl wurde an einem Goldtarget (380 $\mu g/cm^2$) getreut, um die Intensität herabzusetzen. Die Energie der Protonen wurde so bemessen, daß sie in den ersten sieben Detektoren abgebremst wurden und schließlich im letzten Detektor steckenblieben. Auf diese Weise konnte die gesamte Einschußenergie nachgewiesen werden.
Die verwendete Elektronik wird in dem Blockdiagramm Abbildung 7.2 dargestellt. Die Hochspannung wurde durch das C.A.E.N. SY 127 geliefert. Die Detektorspannung betrug 120 V. Bei dieser Spannung sind diese Detektoren vollständigt verarmt.
Das Triggersignal für die ADCs kann wahlweise mit einem der acht Detektoren generiert werden. Bei der hier vorgestellten Messung wurde mit dem achten Detektor getriggert.
Mit dem DAQ-System wurden nun die Daten der acht ADCs ausgelesen und zweifach gespeichert:

* Im *list-mode* auf ein Magnetband und
* im Vielkanal-Mode in acht *displaybuffer* und einem *sumbuffer*.

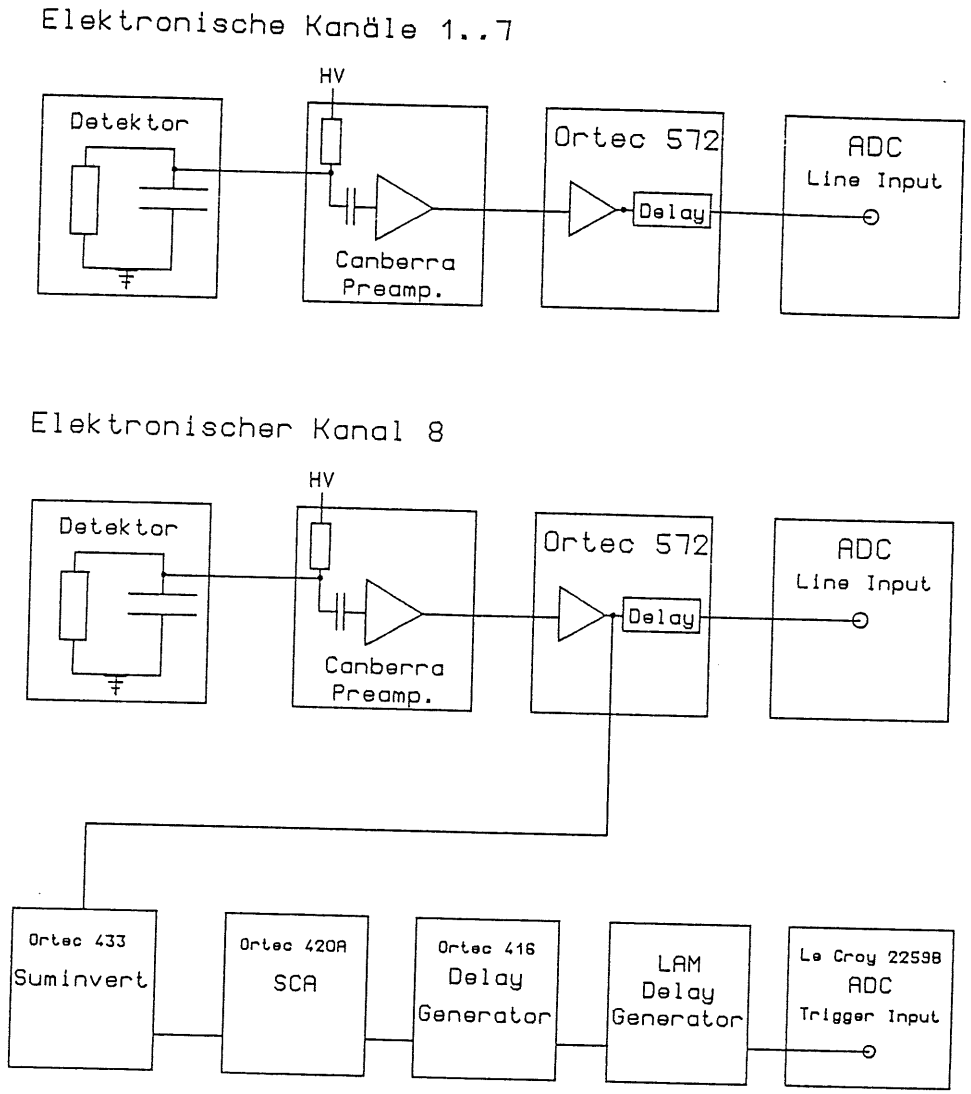


Abbildung 7.2: Blockdiagramm der Elektronik

Mit dieser Versuchsanordnung wurden Meßreihen durchgeführt, die jeweils aus mehreren *runs* bestanden. Ein solcher *runs* wird nun als Beispiel in diesem Kapitel ausgewertet.
Die gemessenen Werte entsprechen den in den Detektoren deponierten Energien.. Für den Vergleich mit theoretischen Vorraussagen ist es günstiger nicht die Energiewerte zu vergleichen, sondern die Detektorstärken d.

7.2 Physikalischer Hintergrund

Zunächst werden die physikalischen Formalismen, die für die Berechnung der theoretischen Werte betrachtet werden müssen, kurz zusammengefaßt. Danach werden die Meßdaten vorgestellt und aus ihnen die Detektordicken berechnet. Diese Werte können Vergleichssdaten gegenübergestellt und bei einer gleichzeitigen Fehlerdiskussion betrachtet werden.
Für die Berechnung der Detektordicken aus der deponierten Energie müssen physikalisch zwei Prozesse berechnet werden.

* Die Streuung der Protonen im Goldtarget und
* der Energieverlust der Protonen im Detektormaterial (Silizium).

Zur Streuung der Protonen im Goldtarget:
Die primären Protonen haben eine Energie von 22.8 MeV. Die Streuung an dem Goldtarget ist eine Coulomb-Streuung und wird durch die bekannte Rutherford-Streuformel beschrieben. Für die Energieberechnung ist jedoch die Intensitätsabhängigkeit des Strahles vom Streuwinkel Θ nicht relevant. Für die Energieabhängigkeit $E(\Theta)$ der abgelenkten Protonen wird ein elastischer Stoß eines Protons mit einem Goldatom betrachtet.
Da Protonen mit einer kinetischen Energie von 22.8 MeV eine Geschwindigkeit von 0.21 c erreichen, werden die Energiewerte sowohl klassisch als auch relativistisch ermittelt. Die analytischen Funktionen $E_{klass}(\Theta)$ und $E_{relativ}(\Theta)$ sind im Anhang C aufgeführt.
In Tabelle 7.1 sind die Ergebnisse zusammengefaßt.

Ruheenergie		Θ	$E_{klass}(\Theta)$	$E_{relativ}(\Theta)$	$\frac{E_{klass}(\Theta)}{E_{relativ}(\Theta)}$
Proton	Au				
[MeV]	[MeV]	[Grad]	[MeV]	[MeV]	[-]
983.3	193000	30	22.76	22.74	1.0009

Tabelle 7.1: Energieverlust des Protons nach einem elastischen Stoß.

Im weiteren wird der Energieverlust in den Detektoren berechnet.
In den Detektoren verlieren die Protonen in 1. Ordnung durch Ionisation Energie. Der Energieverlust pro Wegeinheit (dE/dx) als Funktion der Energie wird durch die Bethe-Bloch Formel ausgedrückt. Die Reichweite der Teilchen L(E) wird bestimmt durch die Integration

$$L(E) = \int_{Eo}^{0} \frac{1}{(dE/dx)} \, dE \, .$$

Dieses Integral ist nur numerisch lösbar und für verschiedene Elemente tabelliert.
Abbildung 7.3 zeigt L(E) für Protonen in Silizium nach einem Tabellenwerk von Williamson [WILL62]. Die Punkte wurden durch die Funktion

$$L(E) = A * (E / MeV)^{b}$$

mit $A = 2.868\ 10^{-3}\ g/cm^2$ und $b = 1.762$
angeglichen. Mit dieser Formel (oder auch mit dem Tabellenwerk) ist es nun möglich, die Dicke d_i des Detektors i aus dem Energieverlust zu berechnen.[2)]

$$d_i = L(E_{i-1}) - L(E_i) \quad , \text{wobei } i = [1,7]$$

und E_0 die Eintrittsenergie ist.
Die Dicke des achten Detektors kann nicht berechnet werden, weil die Protonen ihn nicht vollständig durchfliegen.

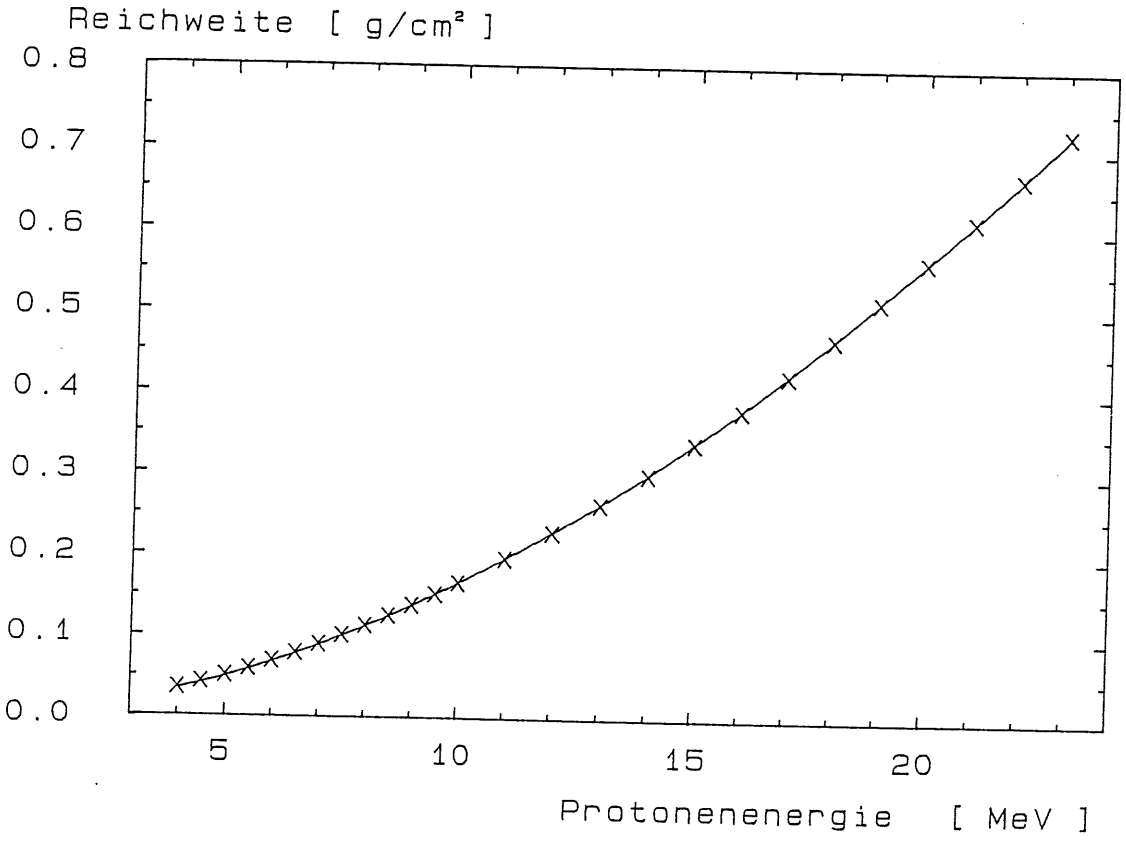


Abbildung 7.3 : Reichweite von Protonen in Silizium.

2) Zusätzlich wird die Dichte von Silizium benötigt. Sie wird von Kittel [KIT83] mit 2.33 g/cm^3 angegeben.

7.3 Experimentelle Daten

Die Spektren, wie sie auch *online* gezeigt wurden, werden in den Abbildungen 7.4 a-j vorgelegt. Im Summenspektrum ist deutlich ein zweiter *peak* bei geringerer Energie zu erkennen, der in den Einzelspektren nicht mehr vom jeweiligen Hauptmaximum getrennt werden kann. Die Ursache des *ghostpeaks* konnte noch während der Strahlzeit eingekreist werden. Der *ghostpeak*[3] entstand durch Verunreinigungen des Targets, d.h., einige Protonen wurden an Kohlenstoffatomen gestreut. Die Energiebilanz verdeutlicht Tabelle 7.2.

Ruheenergie		Θ	$E_{klass}(\Theta)$	$E_{relativ}(\Theta)$	$\frac{E_{klass}(\Theta)}{E_{relativ}(\Theta)}$
Proton	C_{12}				
[MeV]	[GeV]	[Grad]	[MeV]	[MeV]	[-]
983.3	11.2	30	22.36	22.27	1.004

Ohne ein Summenspektrum wäre diese Erscheinung erst bei einer *offline*-Auswertung zu Tage getreten. In der *offline*-Auswertung können die beiden sich überlappenden Spektren durch Setzen eines geeigneten Triggerfensters (siehe DAQOUT im Teil II) separiert werden. Das Ergebnis einer solchen Trennung wird in den Abbildungen 7.5 a-i und Abbildungen 7.6 a-i vorgeführt. (Die Spektrumsform z.B. des *buffers* 1 in der Abbildung 7.5 entsteht durch die Kalibration; siehe auch Kap. 5.2.2.2). Zur genauen Bestimmung der Maxima wurde jeweils ein Gaußfit durchgeführt. Aus diesen Werten wurden die Detektorstärken berechnet. Die Einschußenergie wurde aus den Summenspektren abgelesen. Durch einen Vergleich dieser Energie mit der Summe aus den Einzelspektren läßt sich die Güte der Gaußfits ablesen. Für die Spektren aus den Abbildungen 7.5 a-i ist dieser Fehler innerhalb der *displaybuffer*-Auflösung. Dagegen ist der Fitfehler bei den übrigen Spektren größer.
Die relevanten Werte wurden in den Tabellen 7.3 a-b zusammengefaßt. Die hier angegebenen Fehlerschranken entsprechen dem Auflösungsvermögen des DAQ-Systems bei der Auswertung. Tatsächlich müssen noch andere Fehler berücksichtigt werden (siehe unten).

3) Bei späteren Versuchen wurde das Kalorimeter bei einem Winkel von 45^0 zur Strahlachse positioniert. Auf diese Weise konnten die *peaks* auch in dem Spektrum des 8. Detektors besser getrennt werden und mit dem Single Channel Analyser (SCA, siehe Abb. 7.2) herausgefiltert werden.

Detektor #	Protonenenergie vor Eintritt in den Detektor	Depositum E_i	Det.-Stärke aus der Messung	Det.-Stärke Vergleichsdaten
[-]	[MeV]	[MeV]	[µm]	[µm]
1	22.34	1.71	385 ± 2.2	396 ± 2
2	20.63	1.95	410 ± 2.1	409 ± 2
3	18.68	1.99	386 ± 1.9	381 ± 15
4	16.69	2.25	396 ± 1.7	381 ± 15
5	14.44	2.61	403 ± 1.5	395 ± 2
6	11.83	3.22	411 ± 1.3	381 ± 15
7	8.61	4.79	417 ± 1.0	381 ± 15
8	3.82	3.64	-	395 ± 2
$\sum E_i = 22.16 \pm 0.028$ MeV				

Tabelle 7.3a

Detektor #	Protonenenergie vor Eintritt in den Detektor	Depositum E_i	Det.-Stärke aus der Messung	Det.-Stärke Vergleichsdaten
[-]	[MeV]	[MeV]	[µm]	[µm]
1	22.81	1.71	392 ± 2.2	396 ± 2
2	21.10	1.94	416 ± 2.1	409 ± 2
3	19.16	1.97	391 ± 1.9	381 ± 15
4	17.19	2.22	401 ± 1.8	381 ± 15
5	14.97	2.53	404 ± 1.6	395 ± 2
6	12.44	3.06	411 ± 1.3	381 ± 15
7	9.38	4.23	416 ± 1.0	381 ± 15
8	5.15	5.17	-	395 ± 2
$\sum E_i = 22.83 \pm 0.028$ MeV				

Tabelle 7.3b

Tabelle 7.3: Zusammenfassung der ausgewerteten Meßdaten, (a) für 22.34 MeV Protonen und (b) für 22.81 MeV Protonen.

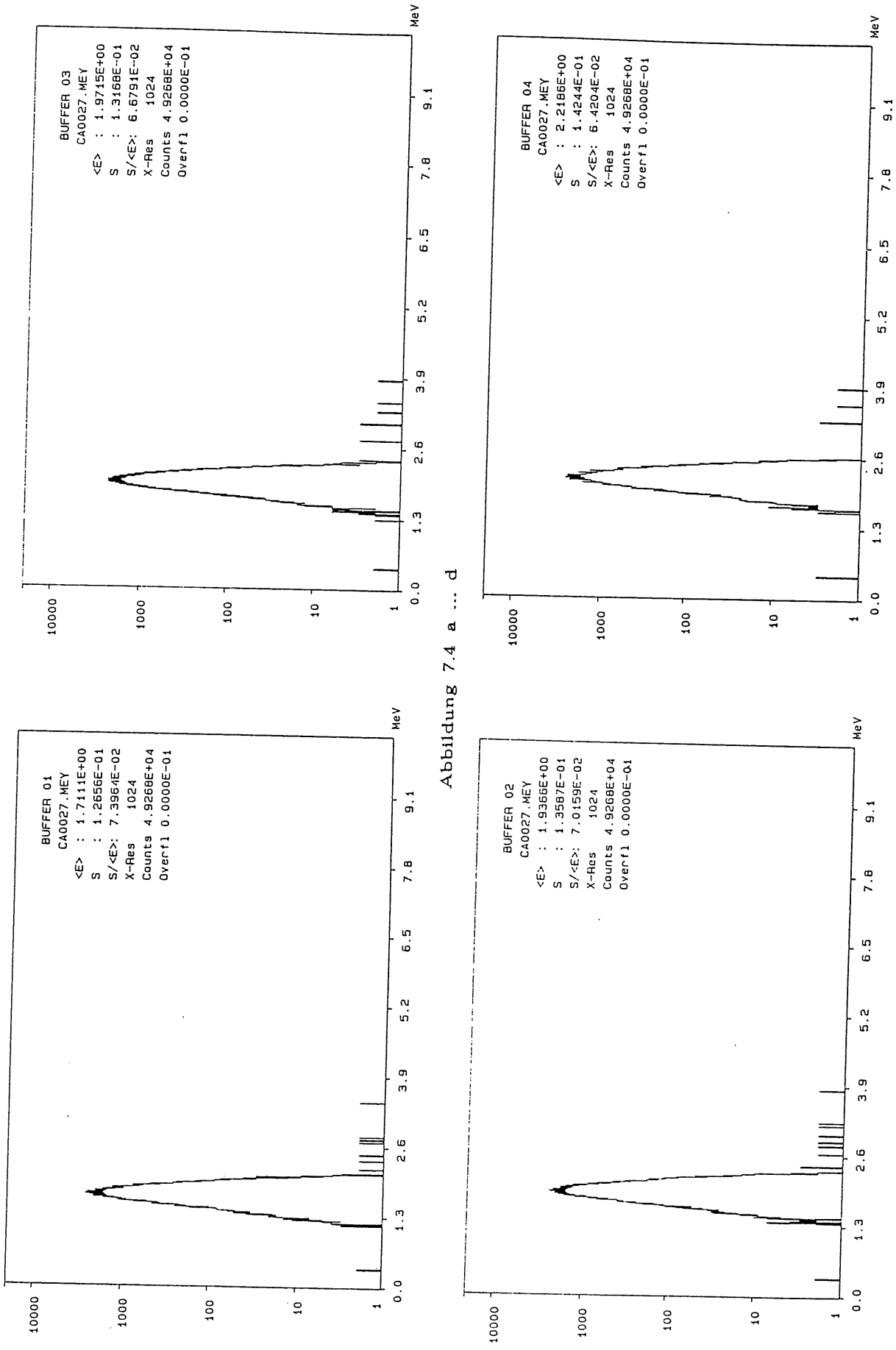


Abbildung 7.4 a ... d

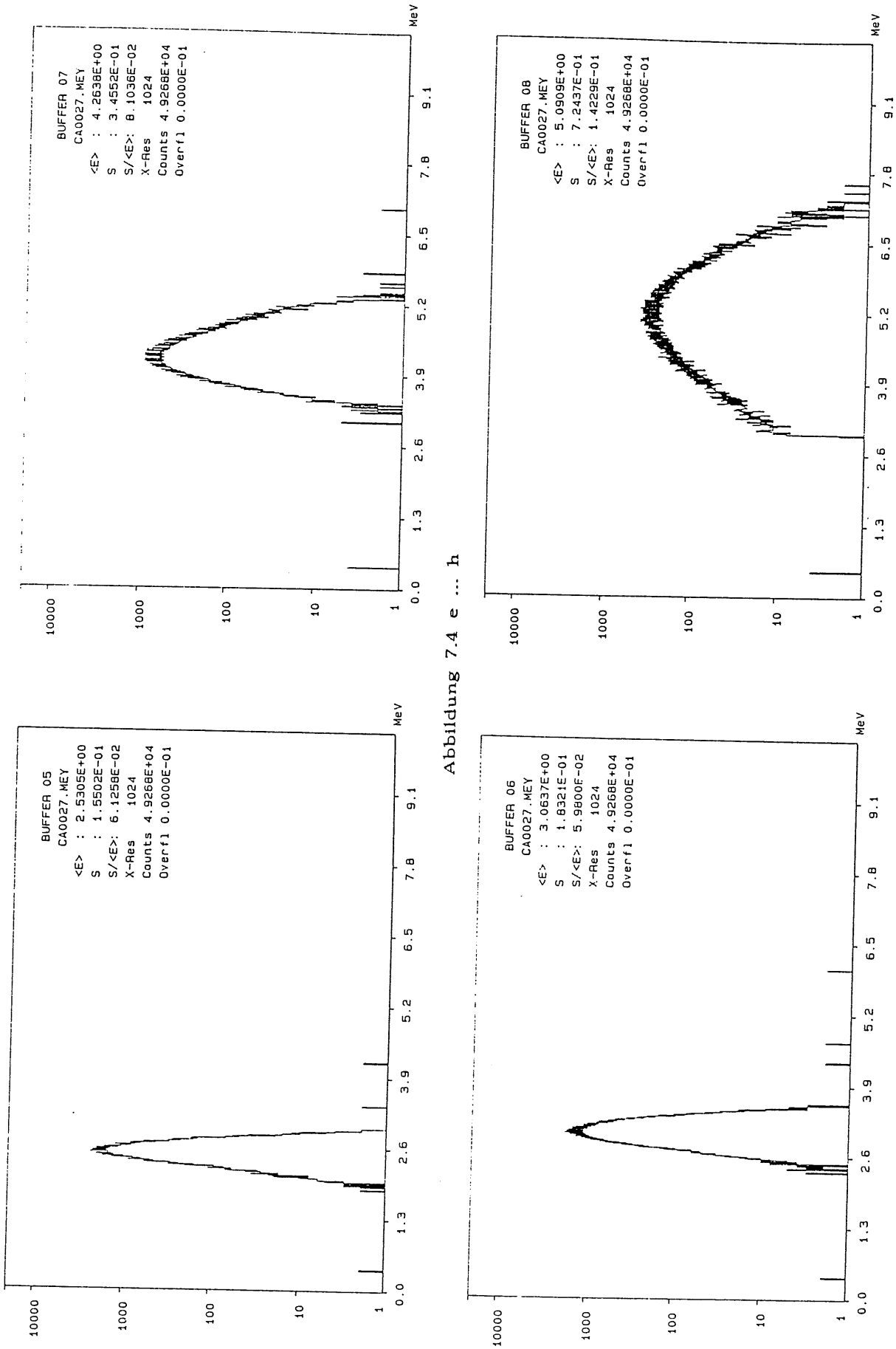


Abbildung 7.4 e ... h

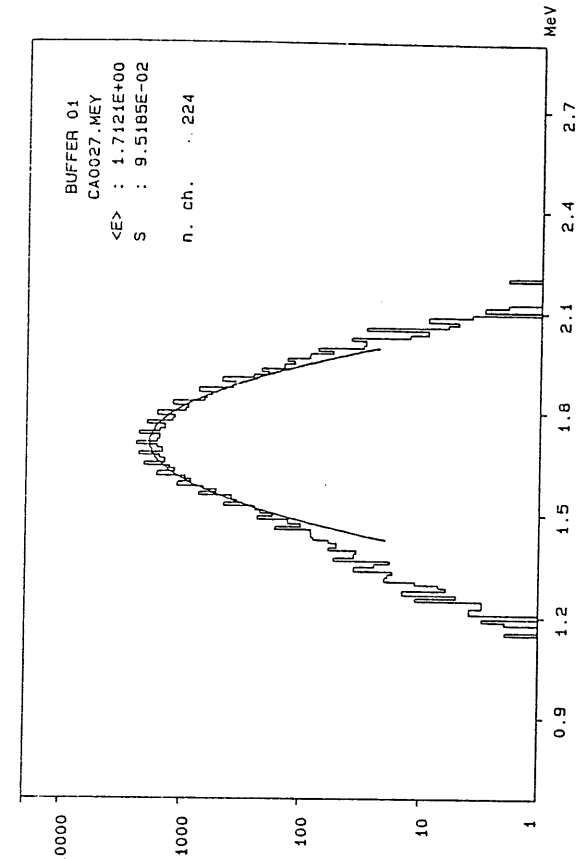
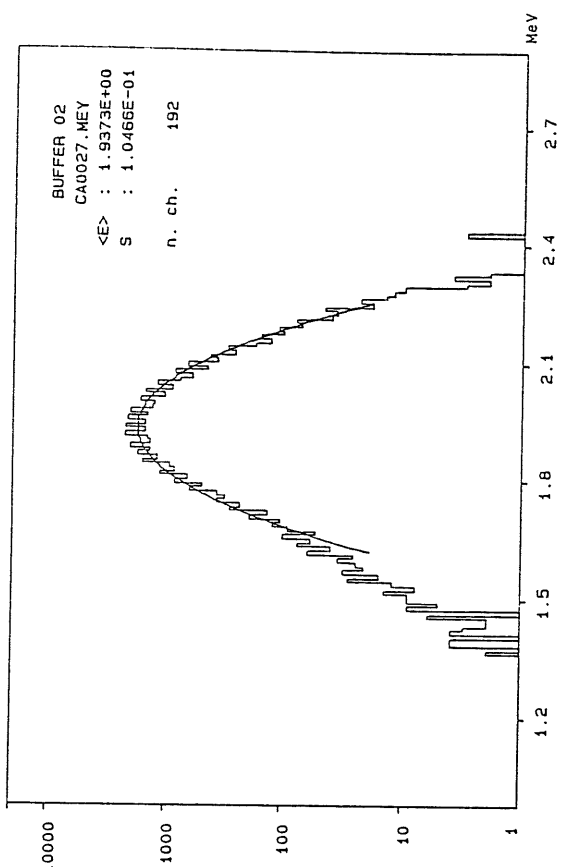


Abbildung 7.5 a ... b



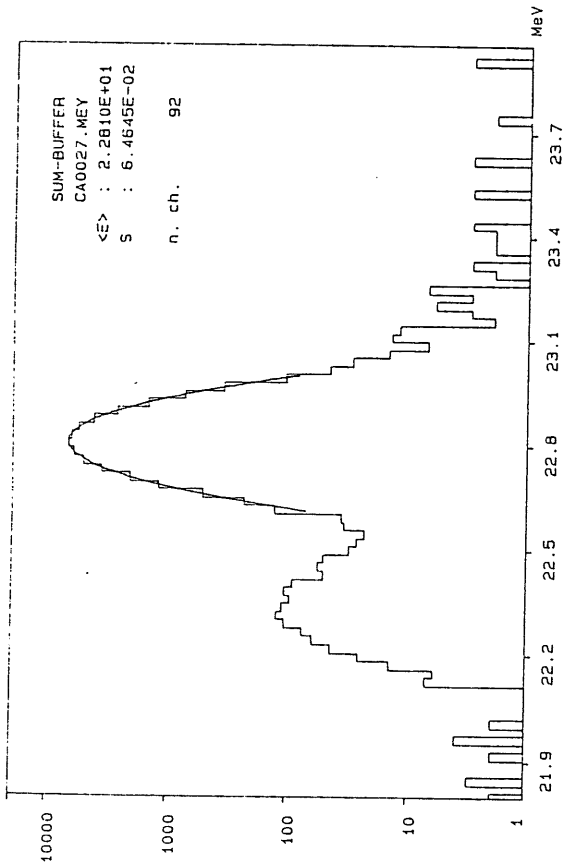
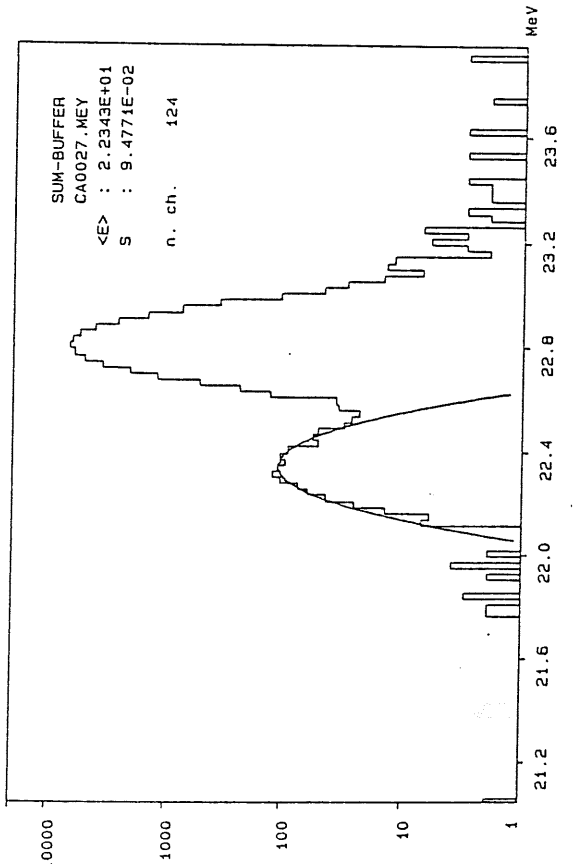


Abbildung 7.4 i ... j



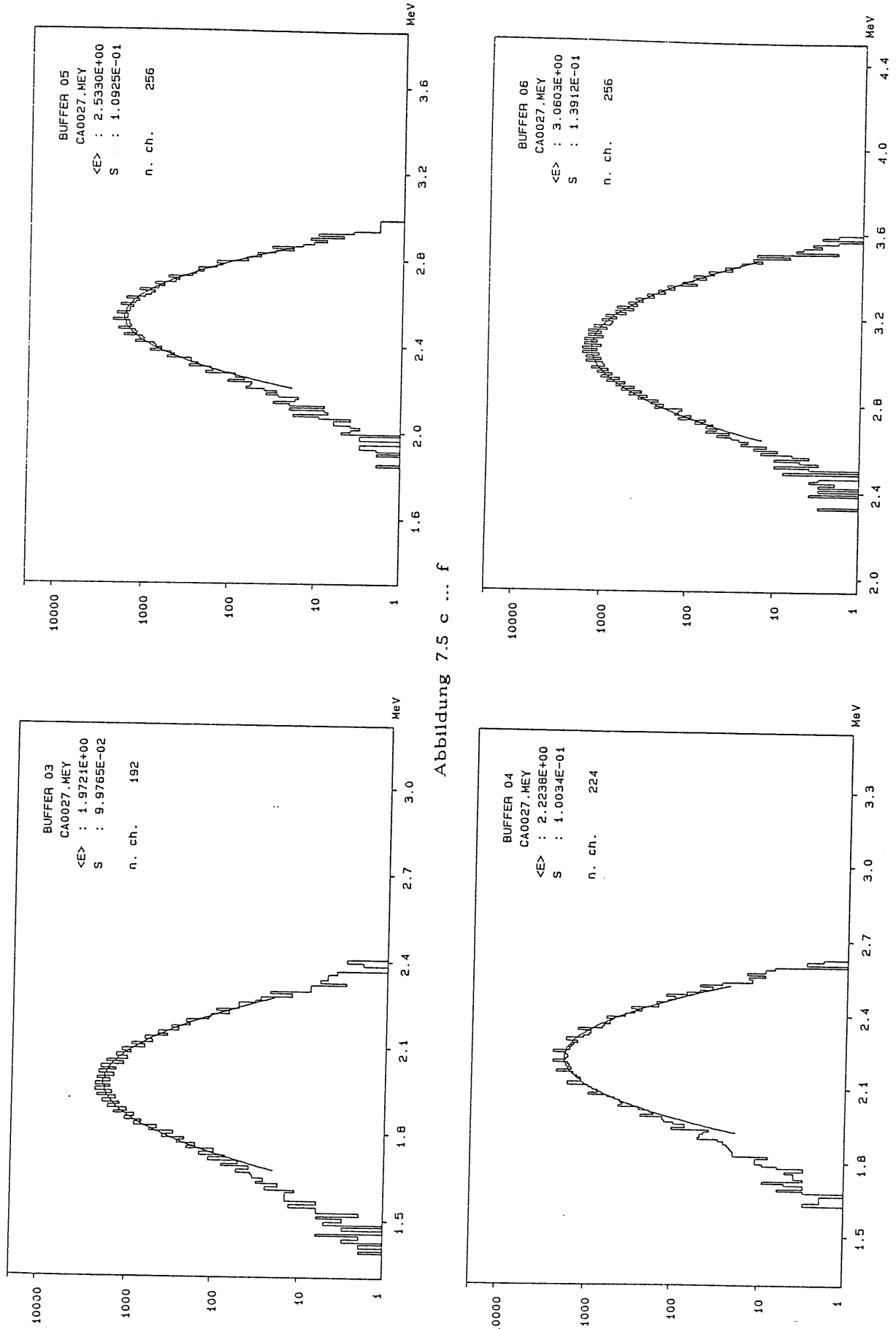
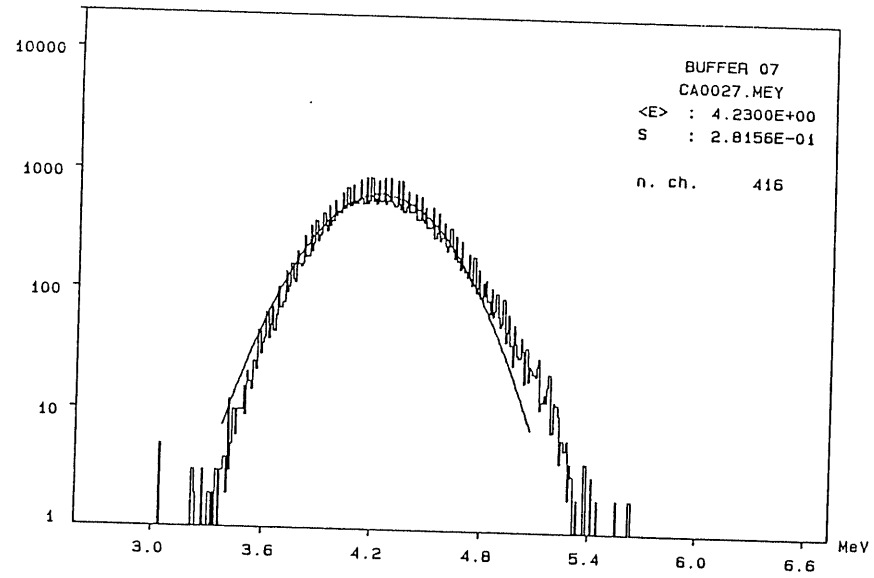


Abbildung 7.5 c ... f



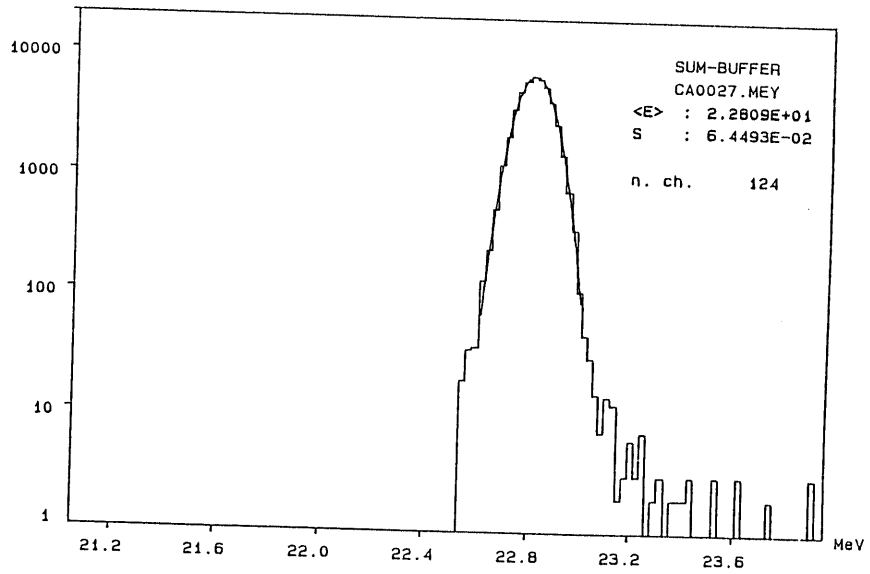
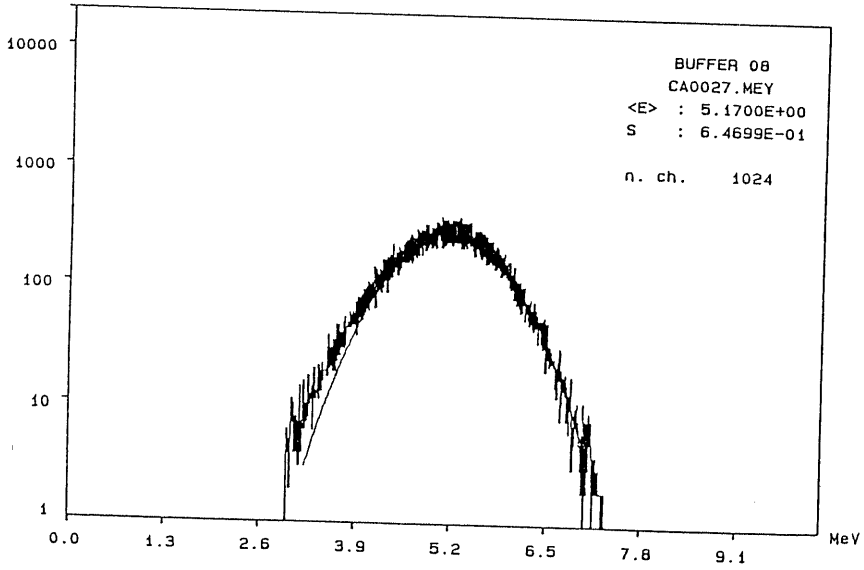


Abbildung 7.5 g ... i



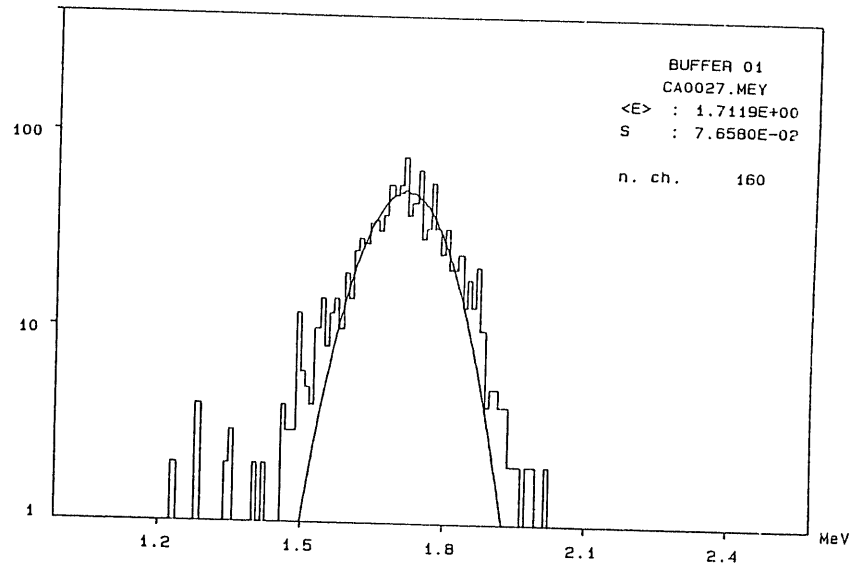


Abbildung 7.6 a

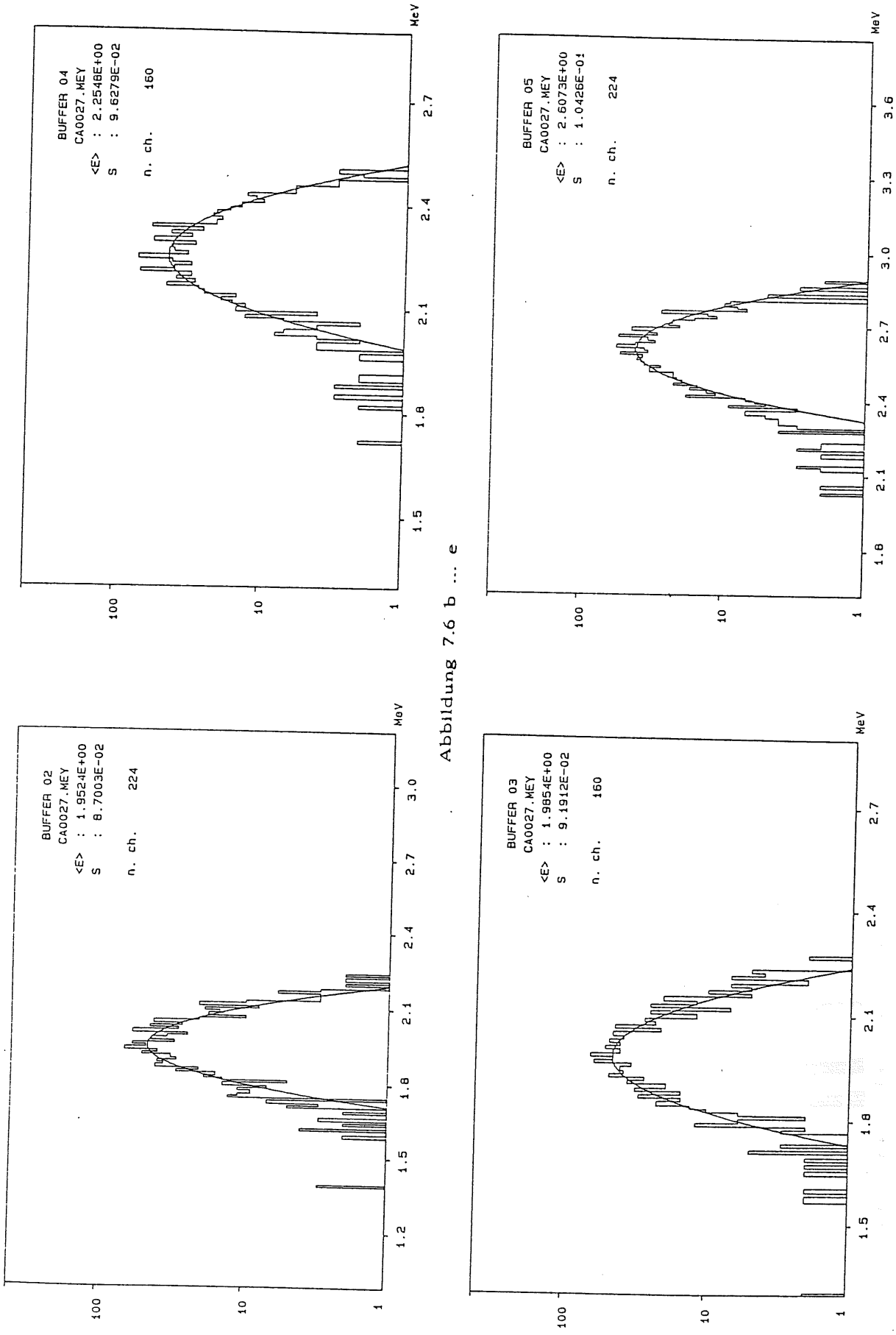


Abbildung 7.6 b ... e

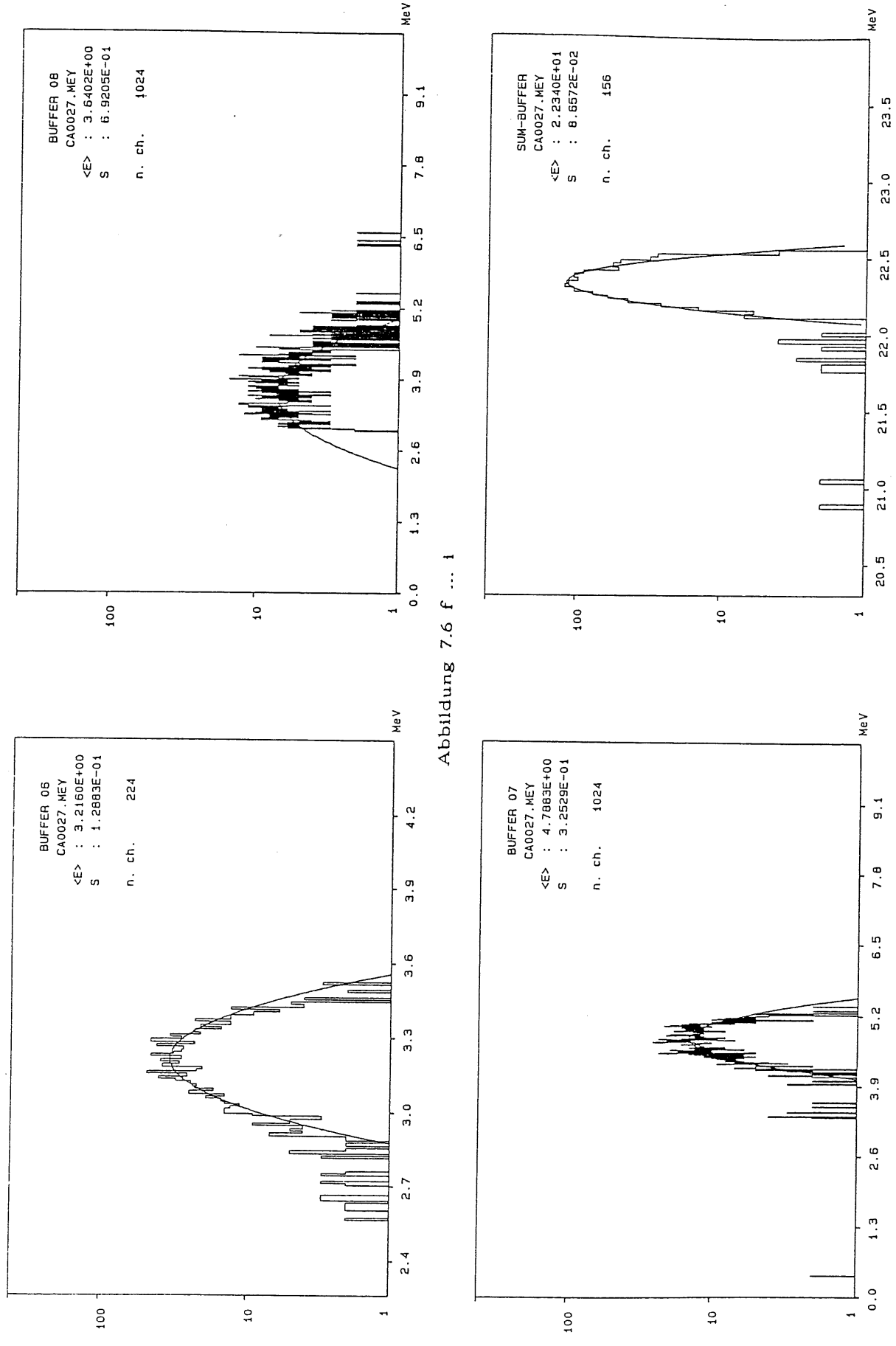


Abbildung 7.6 f ... 1

7.4 Diskussion der Ergebnisse

In diesem Abschnitt soll mit Hilfe einer Fehlerquellenbetrachtung die Qualität der Ergebnisse abgeschätzt werden. Für die Fehlerbetrachtung werden zwei Fehlertypen unterschieden.

* Systematische Fehler und
* statistische Fehler.

Die Systematischen Fehler

* Der Protonenstrahl wurde auf 22.8 MeV justiert, d.h., das Zyklotron wurde entsprechend eingestellt. Die tatsächliche Protonenenergie vor dem Target konnte jedoch nicht bestimmt werden. Hier muß mit einer Abweichung von bis zu + 0.3 MeV gerechnet werden.
* Der Energieverlust der Protonen in den Gold- und Aluminumelektroden der Detektoren beträgt in der Summe ≈ 0.07%.
* Durch die angelegte Detektorspannung werden die Protonen innerhalb der Detektoren beschleunigt und außerhalb um den gleichen Betrag gebremst. Der Effekt ist jedoch vernachlässigbar: ≈ 0.001%
* Es gab während der Messungen Probleme mit der Elektronik. Insbesondere wurde hinter dem zweiten Vorverstärker gelegentlich eine Offsetspannung festgesellt, so daß dann zu hohe Werte angezeigt wurden. Mit diesem Aspekt müssen die Ergebnissen von diesem Detektor betrachtet werden. Eine Fehlerabschätzung wird nicht vorgenommen.
* Zur Erfassung der Drift der Elektronik wurden bei anderen Versuchen Testpulserspektren aufgenommen. Diese zeigten je nach elektronischem Kanal eine Drift zwischen 0.04% - 0.7% in einem Zeitraum von 4 h. Der Zeitabstand zwischen Kalibration und der Messung betrug in diesem Fall
  6 h für die Kanäle 4, 5 und 8,
  12 h für den Kanal 2 und
  30 h für die restlichen Kanäle 1, 3, 6 und 7.
  Es wird deshalb ein Fehler je nach Zeitdifferenz zwischen 0.7% bis 2.0% einkalkuliert.
* Die Auflösung der ADCs betrug 2048 Kanäle mit 4.0 - 4.5 keV pro Kanal.
* Die Auflösung der *displaybuffer* betrug bei der Auswertung 1024 Kanäle und 9.8 keV pro Kanal.
* Die Auflösung des *sumbuffers* betrug bei der Auswertung 1024 Kanäle und 23 keV pro Kanal.
* Die Winkelposition des Kalorimeter war besser als auf 0.1^0 genau einstellbar. Das entspricht einem Fehler in der Eintrittsenergie von 0.07%.

Statistische Fehler

* Die Schärfe des Strahles kann bei der verwendeten Schlitzeinstellung mit ± 13 keV angegeben werden [RUT77].
* Das elektronische Rauschen der Detektoren und der Elektronik betrug je nach Kanal 12 ... 34 keV.

Die statistischen Fehler tragen zur Verbreiterung der nachzuweisenden Summenenergie bei. Nach dem Fehlerfortpflanzungsgesetz wird für die Summenenergie ein $\sigma \approx 59$ keV erwartet.

Aus den σ-Breiten der Einzelspektren kann nicht auf die Breite des Summenspektrums geschlossen werden, da diese Spektren korreliert sind. Durch die Triggerung wird quasi jeweils der Weg eines einzelnen Protons durch die Detektoren verfolgt.

Die Verbreiterung der Einzelspektren vom ersten bis zum letzten Detektor hat im wesentlichen zwei Ursachen:

* Erstens, wird der Strahl durch Vielfachstreuung aufgeweitet. Die Abweichung von der Strahlachse führt dann zu einem längeren Weg durch den Detektor und damit zu einer höheren Energiedeposition.
* Zweitens, verstärkt die Abbremsung der Protonen durch Ionisation eine vorhandene Strahlunschärfe, so daß die langsameren Protonen stärker abgebremst werden als die schnelleren.

Zusammenfassung:

- Innerhalb der Fehlergrenzen wurden die erwarteten Werte nachgewiesen. Dabei haben Verzögerungen zwischen der Kalibration und der Messung zu Problemen geführt. Drifterscheinungen der Elektronik könnten in der Auswertung kompensiert werden, wenn z.B. während der Messung ein Testpulser mitliefe, so daß in den Spektren zusätzliche Referenzpeaks aufgenommen würden. Ein solches Verfahren wäre auch geeignet für die Erkennung von Offsetproblemen wie es der Vorverstärkers Nr. 2 zeigte, zu entlarven.
- Die σ-Breite der Summenenergie stimmt gut mit der Vorhersage überein (65 keV, siehe Abb. 7.5).
- Das DAQ-System hat sich im Experiment voll bewährt. Insbesondere die Notwendigkeit eines energiegeeichten *sumbuffers* wurde hier sehr deutlich.

Kapitel 8

Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Datenaufnahmesystem entwickelt, aufgebaut und getestet, daß speziell für Testexperimente mit Silizium instrumentierte Kalorimeter konzipiert wurde.

Die jetzige Version wird durch folgeden Punkte charakterisiert:

* Bis zu 56 ADCs lassen sich pro *event* auslesen und im *list-mode* auf ein Magnetband speichern. Für die Auslese eines Testkalorimeters ist dies normalerweise ausreichend. Unter bestimmten Voraussetzungen kann die Anzahl der auszulesenen ADCs auf einige hundert erhöht werden.

* Für bis zu 56 ADCs können *displaybuffer* eingerichtet werden mit einer bedingt wählbaren Auflösung zwischen 128 und 2048 Energiekanälen. Die Einträge in diese *displaybuffer* sind energiegeeicht.

* Je nach Anzahl der auszulesenen ADCs kann eine *event*-Rate zwischen 50 - 1000 Hz verarbeitet werden.

* Es wird ein *sumbuffer* mit einer festen Auflösung von 1024 Kanälen eingerichtet, so daß es ebenso möglich ist, während des Experiments das Spektrum der sichtbaren Energie darzustellen.

* Für jeden *buffer* werden Anzahl der *events*, Mittelwert, rms-Wert und Überlaufzähler verwaltet. Hierdurch wird erreicht, daß vor allem auch die mittlere sichtbare Energie und Energieauflösung des Kalorimeters während des Experiments erfaßt wird.

* Mit Hilfe der *lifedisplay*-Funktion kann jedes Spektrum *online* aufgebaut werden. Damit ist die erforderliche Monitorfunktion gewährleistet. Fehler in der Versuchsanordnung können nur so rechtzeitig erkannt und beseitigt werden.

* Spektrumsmanipulation wie Gaußfit, Integral und Lupenfunktion sind möglich.

* Linearer und logarithmischer Maßstab. Der halblogarithmische Maßstab ist insbesondere für die Gaußfitfunktion hilfreich.

* Alle Spektren können für die Dokumentation gedruckt und geplottet werden. Außerdem können sie für eine spätere Auswertung auf der Festplatte oder auf einem Magnetband gespeichert werden.

* Das System schließt die Datenauswertung mit ein, d.h., Auswertungen können sofort vor Ort durchgeführt werden. Zwei Funktionen der Auswerteprogramme seien hier betont:
  a) Bei der Energieeichung der Daten wird auch der nichtlineare Teil der Kalibrationskurven berücksichtigt.
  b) Durch Setzen eines Triggerfensters kann die Funktion eines Single Channel Analyser (SCA) simuliert werden.

* Die Datenbänder sind auch auf Großrechenanlagen auswertbar. Dadurch wird eine maximale Flexibilität in der Wahl des Datenauswertesystems gesichert.

* Es wird hohe Bednutzerfreundlichkeit durch Menüs, weitgehend selbsterklärende Algorithmen und durch Fehlerabfangroutinen erreicht. Außerdem müssen zur Konfiguration des System keine absoluten Speicheradressen eingegeben werden, sondern nur ablesbare CAMAC-Adressen (*slotnumber, input* des ADCs).

Alle Programme sind modular aufgebaut und können durch weitere Module bedingt erweitert werden. Die Erweiterung der schon bestehenden Module, insbesondere der Anzahl der auszulesenen ADCs ist nur bei gleichzeitiger Einschränkung der *displaybuffer*-Auflösung realisierbar.

Zu den Aufgaben der Datenaufnahme und der Datenauswertung wurde das System um die Aufgabe der Spannungseinstellung und -überwachung der Halbleiterdetektoren erweitert. Hierfür wurde das Detektor Current Meter (DCM) und ein zugehöriges Interface (DCI; CAMAC-Einschub) entwickelt.

Das Datenaufnahmesystem hatte sich bereit in Experimenten am Isochronzyklotron erfolgreich bewähren können.

Anhang A

In diesem Anhang werden Informationen gegeben, die insbesondere für Erweiterungen und Änderungen notwendig sind.

A.1 Liste aller Programmroutinen

Die Routinen werden in einer Tabellenform notiert, die sich in der Praxis als sehr hilfreich erwiesen hat. Die Spalten drücken die verschiedenen Overlay-Ebenen aus, die folgende Bedeutung haben:
Der Programmierer ordnet die Routinen eines Programmes mehreren Ebenen, deren Anzahl er festlegt, zu. Dabei gelten für die Overlay-Ebenen folgende Reglementierungen:

* Die 1. Ebene (*Root*) enthält nur eine Routine. Sie wird Hauptprogramm genannt und ist während des gesamten Programmablaufs resident.
* Von den übrigen Ebenen darf jeweils nur **eine** Routine zur Zeit im RAM geladen sein. Das hat zur Konsequenz, daß eine Routine **niemals** eine Routine der gleichen Ebene, sei es direkt oder über eine dritte, aufrufen darf.

Die Lesart der Tabellen wird am Beispiel der folgenden Tabelle gezeigt.
Das Hauptprogramm heißt A37. Die erste Routine, die von A37 aufgerufen wird heißt DATIN1. Von DATIN1 werden 4 Routinen aufgerufen: OPEFI1, ERROR7, DATIN2 und WRTINI. Als zweite Routine wird CHAN37 von A37 aufgerufen, usw.

Root A37 (DAQIN)	
2. Ebene	3. Ebene
DATIN1	OPEFI1, ERROR7, DATIN2, WRTINI
CHAN37	INTASC, SRNIN0, GETCHR, ASCINT, CAMIN2
COMLI1	OPEFI1, DSPIN5, SRNIN1
OPEFI2	
DSDINI	OPEFI1, DSPIN1, DSPIN5, DSPIN2, GETIBF, ERROR7
CALIN1	SRNIN1, GETOFF
SETFAC	OPEFI1, ERROR7
	DSPIN4, SRNIN2

Root A37 (DAQIN) Fortsetzung	
2. Ebene	3. Ebene
GETCST (Kommandoschleife) Befehl	
	SRNIN2, DSPMEN, TIMDSP
"X"	OPEFI1, QITDSP, SRNIN2, WRUTIM, CMCRUN
"T"	TSTBF1, TSTBF2, SRNIN2
"C"	CLRBUF
"F"	OPEFI1, WRTBLK
"L"	SETLOG, SRNIN2
"K"	BLKSET, SRNIN2
"E"	BYEBYE, FILTMP, CLSFIL, OPEFI1, WRTBLK, SAVIBF, AXXEND
"H"	MENUE
"V"	VTAB
"Q"	QITDSP, SRNIN2
"D"	DSPBUF, DSPSUM, DSPBF1
GETCRU (Kommandoschleife) Befehl	
	FILTMP, DSPMEN, TIMDSP QITDSP, SRNIN2, CMCSTP, SRNIN2, ERROR7
"H"	MENUE
"Y"	CMCSTP, YRESOL, CMCRUN
"L"	CMCSTP, SETLOG, SRNIN2, CMCRUN
"S"	CMCSTP, OPEFI1, WSTTIM
"Q"	QITDSP, SRNIN2
"D"	CMCSTP, DSPBUF, DSPSUM, DSPBF1, CMCRUN

Root A37CAL (CALIN)	
2. Ebene	3. Ebene
DATIN3	SRN380, OPEFI1, ERROR, DATIN2, WRTINI
CHANAM	INTASC, SRN380, GETCHR, ASCINT, CAMIN2
COMLI2	OPEFI1, SRNIN1
OPEFI2	
CAMINI	OPEFI1, SRNIN1, CAMIN1, CAMIN3, CAMIN4
	CAMIN2

Root A37CAL (CALIN) Fortsetzung	
2. Ebene	3. Ebene
CALIN2	SRN381, OPEFI1, NEPEAK, CALRQ2
	SRN382
GETCOM (Kommandoschleife) Befehl	
	SRN382, FILTMP, DSCMEN, TIMDSP, QITDSP CMCSTP, ERROR
"X"	QITDSP, SRN382, OPEFI1, WRUTIM, CMCRUN
"C"	CLRBF2
"P"	SRN382, GEHAC1, ERROR, GEHAC2, WRBOBK OPEFI1, NEPEAK, SRN382
"F"	OPEFI1, WRTBLK
"K"	BLKCST, SRN382
"E"	BYEBYE, CLSFIL, OPEFI1, WRTBLK, FILTMP, SAVCBF, AXXEND
"L"	SETLOG, SRN382
"H"	MENUE2
"Y"	YRESOL, SRN382
"Q"	QITDSP, SRN382
"D"	DSCBUF, DSCBF1
"S"	CMCSTP, OPEFI1, WSTTIM
"V"	VTAB
"A"	SRN382, CALRQ0, WRBOBK, ERROR, GEHAC1, GEHAC2, CALRQ1, OPEFI1, SORTCH, DSCBF2, DSPCL2, CUPOSI, SEMARK, SETLIM, DSPCL3, QITDSP, CALFAC, GETCOF, GETCHE, WRTBLK, CLRBF1, SEFA38
	SRN382

Root A39 (GIL)	
2. Ebene	3. Ebene
DATIN5, GETIBF	
DAT390	OPEFI1, ERRO39, AXXEND, CAMIN2, DSPIN5
DAT391	OPEFI1
	ERRO39, SRN391
GETCO3 (Kommandoschleife) Befehl "L" "E" "H" "Y" "V" "Q" "D" "P"	 SETLOG, SRN391 BYEBYE, AXXEND MENUE3 YRESOL VTAB QITDSP, SRN391 DSPBI1, DSPBI2, DSPBI3, DSPMEI PLOT39,DSP390, DSP391, DSP392, INT390, GAUS39
INTEG1 Befehl "I"	 CUPOSI, DSPCNT, SEMARK, INTEGR
MAGNI Befehl "M"	 CUPOSI, DSPCNT, SEMARK, DSPBI1, QITDSP ERRO39, SRN391, DSPBI2, DSPBI3, INTEGR
GAUSS1 Befehl "G"	 CUPOSI, DSPCNT, SEMARK, GAUSS2, GAUSS3, QITDSP, ERRO39, SRN391
PSE390, PSE391 Befehl "S"	 PSE390, PSE391, SRN390

Root A41 (CALOUT)	
2. Ebene	3. Ebene
OPEFI2	
	ERRO41
DAT410	OPEFI1, ERRO41
	SRN410
REHE41	GEHEAD, ERRO41
CAM410	CAM411, CAM412, CAM413, CAM414, CAM415, CAM416, CAM417
	SRN412
GEC410 (Kommandoschleife) Befehl	
	DMEN41, TIMD41, QDSP41, SRN412, ERRO41 STOP41, UPPEVA
"L"	SLOG41, SRN412
"E"	BYEBYE, CLSFIL, WBLK41, SABF41, AXXEND
"H"	MENU41
"Y"	YRES41
"V"	VTAB41
"Q"	QDSP41, SRN412
"D"	DSP410, DSP411
"S"	STOP41
"X"	RUNE41
"P"	CBF410, QDSP41, SRN412, FIEOPE, ERRO41, PMENUE, SORT41, CALRQ1, DSP412, DSP413, CUPOSI, SEMARK, SELI41, DSP414, QDSP41, CAFA41, GETCOF, GETCHE, SEFA41, CBF411, SHDABK, PRDABK
"C"	CBF410
"K"	BLKS41, SRN412
CORPEV	SRN411, CPV411, CPV412
ACC410	SRN411, ACC411, ACC412

Root A42 (COPYTD)	
2. Ebene	3. Ebene
SRN420	
	DAT420
CHAN42	OPEFI1, ERRO42, GETCHR, ASCINT
COPTTD	
	EXUNHO

Root A43 (COPYDT)	
2. Ebene	3. Ebene
SRN430	
DAT430	
COPDTT	
ERRO43	
EXUNHO	

Root A44 (SAVPAS)	
2. Ebene	3. Ebene
	DAT440, SRN440, SRN442
GEC440	MENU44
CHA440	OPE441, ERRO44, SRN440, GETCHR, OPE440
CHA441	OPE441, ERRO44, SRN440, GETCHR, OPE442
CHA442	OPE443, ERRO44, SRN440, GETCHR, OPE444
CHA443	OPE443, ERRO44, SRN440, GETCHR, OPE445

Root A40 (DAQOUT)	
2. Ebene	3. Ebene
	OPEFI2, ERRO40
DAT400	OPEFI1, ERRO40
	SRN400
REHEAD	GEHEAD, ERRO40
	ERRO40
DIN400	SRN401, GEBF40
DIN401	ERRO40, DIN402, DIN403
CAL400	SRN401, GETOFF
SEFA40	ERRO40
	SRN402
GEC400 (Kommandoschleife) Befehle "X" "C" "K" "E" "V" "L" "H" "Y" "Q" "D" "S" "P"	 RUNE40 CBF400 BLKS40, SRN402 BYEBYE, CLSFIL, WBLK40, SABF40, AXXEND VTAB40 SLOG40, SRN402 MENU40, MEN240 YRES40 QDSP40, SRN402 DSP40, DSP402, DSP401 STOP40 DPP400, DPP401, DPP402, DPP403, PIN400 PGAU40
INT400 "I"	CUPO40, DCNT40, SEMA40, INT401
MAG400 "M"	CUPO40, DCNT40, SEMA40, DPM400, DPM401 DPM402, QDSP40, ERRO40, SRN402, INT401
GAU400 "G"	CUPO40, DCNT40, SEMA40, GAU401, GAU402 QDSP40, ERRO40, SRN402
TRIG40 "N"	CUPO40, DCNT40, SEMA40, QDSP40, ERRO40 SRN402

Root A40 (DAQOUT) Fortsetzung		
2. Ebene	3. Ebene	4. Ebene
GEC401 (Kommandoschleife) Befehle "Q" "D" "S"	 QDSP40, SRN402 DSP40, DSP402, DSP401 STOP40	CALSUM, CALCHN

A.2 Liste der wichtigsten globalen Variablen

ADCRES	Auflösung der ADCs.
BLOCK	Blockzähler für A37LST.TMP.
BUFNUM	Index des Feldes IADCMC.
CHANN	Eindimensionales Feld; beinhaltet die Mittelwerte der *peaks* bei der Kalibration.
CMRADD	Adresse des "controll and status register" des Crate Control.
COFACT	Kalibrierungskoeffiezienten.
DELAY	Konvertierungszeit der ADCs.
DSPCON	Zweidimensionales Feld, beinhaltet N, $\sum x_i$, $\sum x_i^2$ für alle *displaybuffer* und für den *sumbuffer.*
ERRLOG	Fehlernummer.
HIEOLD	Höchster Energiewert der *displaybuffer.*
HIESUM	Höchster Energiewert des *sumbuffers.*
IADCMC	Eindimensionales Feld, beinhaltet die Adressen der ADCs, die im *list-mode* gespeichert werden sollen.
IBUFF	Eindimensionales Feld; beinhaltet alle *diplaybuffer* und den *sumbuffer.*
IDISPL	Eindimensionales Feld; beinhaltet die Adressen der ADCs, die für ein *displaybuffer* bestimmt worden sind.
IDRESO	Auflösung der *displaybuffer.*
IDSPL	Nummer des *displaybuffers*, das auf dem Graphikterminal momentan angezeigt wird.
IPENUM	*peak*-Nummer; Index von CHANN und SCALE.
IYRESO	Auflösung der Y-Achse der Spektren (*Counts*).
LAMADD	Adresse des LAM-Generators im CAMAC-Rahmen.
LOGON	Logarithmischer Maßstab *On/off.*

NUMADC	Anzahl der ADCs, die pro *event* im *list-mode* gespeichert werden.
NUMDSP	Anzahl der eingerichteten *displaybuffer.*
OFFOLD	Nullstelle auf der *displaybuffer*-Kanalachse.
OFFSET	Kalibrierungskoeffiezienten.
OVERCN	Überlaufzähler.
PROID	Identität des Programms, das als letztes Programm Daten von DISPLY.DAT verändert hat.
SCALE	Eindimentionales Feld; beinhaltet die Sollwerte der *peaks* bei der Kalibration.
TMPBUF	Temporärer Buffer für die *list-mode*-Daten.

A.3 Datenstrukturen der externen Dateien

Die Daten sind in Blöcken zu je 512 Byte aufgeteilt.

A.3.1 DISPLY.DAT

Diese Datei ist 7 Blöcke groß.

Variablenname	Größe	Position im Block	Block
	[Byte]	[Byte]	[Block]
IDRESO	2	0	1
IDISPL	112	2	1
IDSOFF	112	124	1
-	20	236	1
IADCMC	112	256	1
OFFSET	114	368	1
-	30	482	1
COFACT	228	0	2
BLOCK	2	230	2
HIEOLD	4	232	2
OFFOLD	2	236	2
HIESUM	4	238	2
-	14	242	2
RUNNUMBER	2	256	2
RUNNAME (RAD50)	8	258	2
RUNNAME (ASCII)	10	266	2
STARTTIME (des *runs*)	20	276	2
STOPTIME (des *runs*)	20	296	2
PROID	2	316	2
-	194	318	2
COMMENT	240	0	3
HEADID	10	240	3
-	6	250	3
IPENUM	2	256	3
CHANN	124	258	3
SCALE	124	382	3
-	6	506	3

Parameter der *Offline*-Auswertung

Variablenname	Größe	Position im Block	Block
	[Byte]	[Byte]	[Block]
OFFSET	224	0	4
-	32	224	4
COFACT	224	256	4
-	32	480	4
A	224	0	5
-	32	224	5
B	224	256	5
-	32	480	5
C	224	0	6
-	32	224	6
D	224	256	6
-	32	480	6
TRANS	112	0	7
IADC	112	112	7
-	32	224	7
HIEOLD	4	256	7
OFFOLD	2	260	7
HIESUM	4	262	7
NUMADC	2	266	7
RUNNAME (*last*)	10	268	7
-	234	278	7

A.3.2 IBUFF.DAT

Diese Datei ist 66 Blöcke groß.

Variablenname	Größe	Position im Block	Block
	[Byte]	[Byte]	[Block]
DSPCON	512	0	1
DSPCON (Fortsetzung)	172	0	2
-	340	172	2
IBUFF		0	3
(30720 Byte = 60 Blöcke)			62
TMPBUF		0	63
(2048 Byte = 4 Blöcke)			66

A.3.3 A37LST.TMP

Die *list-mode*-Daten werden in dieser temporären Datei auf der Festplatte gespeichert. Für die entsprechende Kopie auf dem Magnetband sind zwei Besonderheiten zu beachten.

* Die Datei auf dem Magnetband hat einen anderen Namen (*run-name*).
* Die Kopie ist um den sogenannten *header*-Block erweitert.

Die kleinste Einheit, in denen die *list-mode*-Daten, vom RAM auf die Festplatte transferiert werden, umfaßt 4 Blöcke (siehe Kap. 3.2). Eine solche Einheit wird *event*-Blockeinheit genannt. In dem Datenaufnahmeprogramm CALIN werden außerdem auch noch "Informationsblöcke" geschrieben (siehe Kap. 3.3.2). Damit ein Auswerteprogramm diese drei Blöcke bzw. Einheiten unterscheiden kann, wurden folgende Definitionen vereinbart.

* Der *header*-Block ist der erste Block der *run*-Datei.
* Das höchste Bit eines Datenwortes (16 Bit) dient als Unterscheidungsmerkmal zwischen Daten- und Kontrollworten. Das bedeutet, daß ein Datenwort in dem Bereich von 0 ... 32767 (2^{15} - 1) liegt und ein Kontrollwort entsprechend in dem Bereich von -32768 ... -1 liegt. In dieser Version werden nur zwei Kontrollwörter verwendet.
  * EOE (*end of event* = -1); Dieses Kontrollwort wird zwischen den ADC-Daten zweier *events* geschrieben. Außerdem wird mit ihm der Rest einer *event*-Blokkeinheit "aufgefüllt", wenn der Platz für alle Daten eines weiteren *events* nicht ausreichend sein sollte.
  * IFO (Informationsblock = -2); Dieses Kontrollwort steht immer am Anfang und am Ende eines Informationsblocks.

Die Datenstruktur dieser Blöcke bzw. Blockeinheiten werden im folgenden gezeigt.

Header-Block

Variablenname	Größe	Position im Block	Block
	[Byte]	[Byte]	[Block]
Versionsnummer	10	0	1
-	70	10	1
RUNNAME	10	80	1
STARTTIME	20	90	1
STOPTIME	20	110	1
-	30	130	1
COMMENT	240	160	1
-	112	400	1

Event-Dateneinheit

Die Struktur wird an einem Beispiel am besten deutlich. Es wird angenommen, daß pro *event* 2 ADCs ausgelesen werden.

Variablenname	Größe [Byte]	Position im Block [Byte]	Block [Block]
Datenwort des 1. ADCs	2	0	i
Datenwort des 2. ADCs	2	2	i
EOE	2	4	i
Datenwort des 1. ADCs	2	6	i
Datenwort des 2. ADCs	2	8	i
EOE	2	10	i
.	.	.	.
.	.	.	.
.	.	.	.
Datenwort des 1. ADCs	2	504	i
Datenwort des 2. ADCs	2	506	i
EOE	2	508	i
Datenwort des 1. ADCs	2	510	i
Datenwort des 2. ADCs	2	0	i + 1
EOE	2	2	i
.	.	.	.
.	.	.	.
.	.	.	.
Datenwort des 1. ADCs	2	508	i + 1
Datenwort des 2. ADCs	2	510	i + 1
EOE	2	0	i + 2
.	.	.	.
.	.	.	.
.	.	.	.
Datenwort des 1. ADCs	2	506	i + 2
Datenwort des 2. ADCs	2	508	i + 2
EOE	2	510	i + 2
.	.	.	.
.	.	.	.
.	.	.	.
Datenwort des 1. ADCs	2	0	i + 3
Datenwort des 2. ADCs	2	2	i + 3
EOE	2	4	i + 3
.	.	.	.
.	.	.	.
.	.	.	.
Datenwort des 1. ADCs	2	504	i + 3
Datenwort des 2. ADCs	2	506	i + 3
EOE	2	508	i + 3
EOE	2	510	i + 3

Informationsblock

Der Informationsblock umfaßt nur 256 Bytes. Deshalb beginnt er entweder am Anfang (Byte 0) oder in der Mitte (Byte 256) eines 512 Bytes großen Blocks. Die übrigen 256 Bytes des Blocks werden für ADC-Daten verwendet.

Variablenname	Größe	Position im Block	Block
	[Byte]	[Byte]	[Block]
IFO	2	0	j
BUFNUM (Index von IADCMC)	2	2	j
N (*slotnumber* im Crate)	2	4	j
A (Subadresse des *slots*)	2	6	j
SCAFAC Skalenfaktor [keV/Skt]	4	8	j
SCAVAL Sollwert [MeV]	4	12	j
DSPCON (1,1) (*Counts*)	4	16	j
DSPCON (1,2) ($\sum x_i$)	4	20	j
DSPCON (1,3) ($\sum x_i^2$)	4	24	j
BLOCK	2	28	j
CALABO (*Abort calibration:* 0 = valid calibration, -1 = aborted calibration)	2	30	j
IFO	2	32	j
.	.	.	.
.	.	.	.
.	.	.	.
IFO	2	252	j
IFO	2	254	j

Anhang B

Test der Software

Die Tests sind tabellarisch zusammengestellt. Die Tabelle ist folgendermaßen zu verstehen:

* In der ersten Spalte stehen die Programme, mit denen ein Einzeltest durchgeführt worden ist.
* In der zweiten Spalte steht die zu testende Eingabevariable.
* In Spalte drei sind die tatsächlichen Eingabewerte notiert.
* In der Kommentarspalte sind die zu erwartenden Reaktionen vermerkt.

Programm	Variable	Testwert	Kommentar
DAQIN, CALIN	Cratenumber	1,2,3,4	Wenn der eingegebene Wert nicht mit der Schalterstellung des Cratecontrollers übereinstimmt, so folgt ein Programmabsturz, wenn auf den Crate Controller zugegriffen wird.
DAQIN	Delay	1..255	siehe Kapitel 7
DAQIN, CALIN	Runname	z.B. RU1111.HRZ	Der Name wird in die erste Monitorzeile geschrieben.
DAQIN, CALOUT	Comment	drei Zeilen	Der eingegebene Kommentar muß in den Auswerteprogrammen DAQOUT bzw. CALOUT wieder erscheinen.
DAQIN, CALIN	Command	"X"	1.) Es wird die Startzeit abgespeichert. Sie muß in den Auswerteprogrammen wieder erscheinen. 2.) CAMAC wird gestartet. Der Status "CAMAC started" wird in der letzten Monitorzeile angezeigt.

Programm	Variable	Testwert	Kommentar
DAQIN, CALIN	Command	"S"	1.) Es wird die Stopzeit abgespeichert. Sie wird in den Auswerteprogrammen wiedergegeben. 2.) Der Status "CAMAC stopped" wird auf dem Monitor angezeigt.
DAQIN, CALIN, DAQOUT, CALOUT	Command	"K"	Der aktuelle Wert des Blockzählers wird angezeigt, so daß ein zurücksetzen durch nochmaliges Aufrufen kontrolliert werden kann.
DAQIN, CALIN, DAQOUT, CALOUT, GIL	Command	"L"	"D" zeigt das Spektrum in dem gewählten Maßstab. Bei "X" müssen die Einträge in das Spektrum im richtigen Maßstab vorgenommen werden.
DAQIN, CALIN	Command	"E"	Bevor DAQIN oder CALIN beendet werden, werden eine Reihe von Parametern auf den Plattenspeicher gerettet. Die Parameter können im einzelnen nachgeprüft werden: Die konfigurierten ADC-Adressen (für die *list-mode*-Speicherung), die Kalibrationsparameter und der Blockzähler können wieder in CALIN gesichtet werden. Die konfigurierten ADC-Adressen (für die Speicherung in den Spektren), die *displaybuffer*-Auflösung und die Parameter des Energiefensters können in DAQIN besehen werden. Der Runname, die Start- und Stopzeit und der Kommentar werden in DAQOUT bzw. in CALOUT wiedergegeben. Der freie Raum des *eventbuffers* wird mit EOE-Marken ("-1") aufgefüllt. Dies kann durch Inspektion der temporären Datei "DU1:A37LST.TMP" nachgeprüft werden. Hierzu wird die Betriebssystemroutine "DUMP" verwendet. Das gespeicherte Spektrum kann durch GIL geladen werden.

Programm	Variable	Testwert	Kommentar
DAQIN, CALIN, DAQOUT, CALOUT, SAVPAS, GIL	Command	"H"	Das Menü erscheint prompt.
DAQIN, CALIN, DAQOUT, CALOUT, GIL	Command	"Y"	Kontrolle durch "D".
DAQIN, CALIN, DAQOUT. CALOUT, GIL	Command	"Q"	Das Spektrum verschwindet.
DAQIN, CALIN, DAQOUT, CALOUT, GIL	Command	"D"	Das entsprechende Spektrum wird gezeigt.
DAQIN, CALIN, DAQOUT, CALOUT, GIL	Command	"V"	Der Drucker schiebt das Papier einige Zeilen vor.
CALIN, CALOUT	Command	"F"	Die neu eingestellten Adressen werden prompt geechot. Ob auch das richtige ADC ausgelesen wird, kann durch die Kalibration dieses ADCs nachgewiesen werden. Außerdem können die Kalibrationparameter angezeigt oder verändert werden. Jede Veränderung wird prompt angezeigt.

Programm	Variable	Testwert	Kommentar
DAQIN, DAQOUT	Command	"F"	Das Energiefenster kann durch "D" kontrolliert werden. Die korrekte Energie-Kanal-Zuordnung kann mit einem Testpulser gezeigt werden: Bei gleicher Testpulsereinstellung müssen unterschiedliche Energiefenster innerhalb der Fehlertoleranz (siehe Kap. 5) zum gleichen Mittelwert führen.
DAQIN, CALIN, DAQOUT, CALOUT	Command	"C"	Die Kanaleinträge des entsprechenden *displaybuffers* werden gelöscht. Das Resultat wird durch "D" sichtbar gemacht.
CALIN	Command	"A", "P"	1.) Es wird ein *peak*-Parameterblock geschrieben, der in CALOUT wieder ausgelesen wird. 2.) Es werden die Koeffizienten für die lineare Eichung berechnet. Die Richtigkeit dieser Berechnung wird mit einem "Zyklus-Test" (siehe unten) gezeigt.
DAQIN	Command	"B"	Drei Dinge werden hier vereinbart: 1.) Die neuen Adressen werden sofort angezeigt. Mit "X" und "D" kann geprüft werden, ob der richtige ADC ausgelesen wird. 2.) Die *displaybuffer*-Auflösung wird durch "D" sichtbar gemacht. 3.) Eventuell geladene Spektren werden durch "D" gezeigt.
DAQIN	Command	"T"	Die Test-Spektren werden durch "D" sichtbar gemacht.
GIL, SAVPAS	Command	"E"	Das Programm wird beendet.
GIL, DAQOUT	Command	"I", "M"	Durch geeignete Spektren können diese Funktionen kontrolliert werden.
GIL, DAQOUT	Command	"G"	Der Gaußfit kann an einem Testspektrum, das mit DAQIN ("T") erzeugt wurde, geprüft werden.

Programm	Variable	Testwert	Kommentar
GIL	Command	"S", "P"	Diese Funktionen werden durch einen Plotter prompt ausgeführt.
DAQOUT, CALOUT	Command	"X"	Die Auswertung der *list-mode*-Daten wird gestartet. Als Echo erscheint "evaluation run" in der letzten Monitorzeile.
DAQOUT, CALOUT	Command	"S"	Die Auswertung wird gestoppt. Dieser Status wird ebenfalls angezeigt.
DAQOUT, CALOUT	Command	"E"	Bevor die Programme beendet werden, werden mehrere Parameter auf die Festplatte kopiert. 1.) Die Spektren können in GIL bzw. DAQOUT wieder geladen werden. 2.) Die Kalibrationkoeffizienten können in CALOUT besichtigt werden. 3.) Die Parameter des Energiefensters können in DAQOUT kontrolliert werden.
DAQOUT	Command	"B"	Drei Vereinbarungen werden getroffen: 1.) Die korrekte Zuordnung von *list-mode*-Daten zu den Einzelspektren wird durch den Zyklus-Test gezeigt. 2.) Die *displaybuffer*-Auflösung kann durch "D" nachgeprüft werden. 3.) Eventuell geladene Spektren werden durch "D" angezeigt.
DAQOUT	Command	"T", "P"	Diese Plotfunktionen werden durch einen Plotter prompt ausgeführt.
DAQOUT	Command	"N"	Die Wirkung des Software-Triggers wird in den Spektren bei "X" sichtbar.
CALOUT	Command	"P"	Diese Funktionen werden in dem Zyklus-Test geprüft.
COPYDT, COPYTD, CMPTWD	Datenbytes	-128..127	Diese Programme werden in dem Zyklus-Test geprüft.

Programm	Variable	Testwert	Kommentar
SAVPAS	Command	"L", "D"	Hier wird die Parameterdatei geladen bzw. kopiert. Die einzelnen Parameter können in den Programmen überprüft werden: Für die *offline*-Auswertung werden in CALOUT die Kalibrationsparameter, in DAQOUT die Parameter für das Energie fenster und die Auslesekonfiguration (siehe "B") nachgeprüft. Für die *online*-Auswertung werden in DAQIN die *displaybuffer*-Auflösung, der Blockzähler, die ADC-Konfiguration für die *displaybuffer*, die Fensterparameter, in CALIN die ADC-Konfiguration für die *list-mode*-Daten und die Kalibration koeffizienten nachgeprüft.
SAVPAS	Command	"T", "S"	Hier werden die Spektren geladen bzw. kopiert. Die Spektren können von den Programmen GIL, DAQOUT oder DAQIN wieder geladen werden.
INIFIL	Platte DU1		Die Initialisierung der Datei A37LST.TMP kann mit Hilfe der Systemroutine DIR überprüft werden.
DAQIN, CALIN	ADC-Wert	0..2047	Der ausgelesen Wert wird 1.) in den temporären Speicher geschrieben und kann durch die Systemroutine DUMP inspiziert werden, 2.) in ein Spektrum eingetragen, das durch DUMP (IBUFF.DAT) gesichtet werden kann und 3.) zur Bildung von Mittelwert, Standard abweichung etc. verwendet, die durch "D" angezeigt werden.

Zyklus-Test

Der Zyklus besteht aus vier Schritten:

* Kalibration mehrerer ADCs mit CALIN.
* Aufnahme einiger geeigneter Spektren mit dem Testpulser durch DAQIN.
* Auswertung der Kalibrations-*runs* durch CALOUT.
* Auswertung der übrigen *runs* durch DAQOUT.

Schlußfolgerungen aus dem Zyklus-Test

Mit den ersten beiden Schritten kann die *online*-Kalibration nachgeprüft werden. Mit den definierten Spektren können die Größen für Mittelwert, Standardabweichung, etc. kontrolliert werden.

Mit Schritt 3 werden die Funktionen von

* COPYDT,
* COPYTD,
* CMPTWD und
* CALOUT, Menüpunkt "P"

überprüft.

Mit Schritt 4 wird schließlich die *offline*-Kalibration (CALOUT Menüpunkt "A") und der Menüpunkt "B" des Programms DAQOUT geprüft.

Anhang C

Kinematik

Es soll der Fall betrachtet werden, in dem ein Teilchen 1 mit der Masse m_1 elastisch auf ein ruhendes Teilchen 2 mit der Masse m_2 stößt. Gesucht wird dabei Restenergie, die das Teilchen 1 nach dem Stoß in Abhängigkeit vom Ablenkwinkel Θ hat. Zur Berechnung der Funktion $E(\Theta)$ wird der klassische und der relativistische Fall betrachtet.

C.1 Der klassische Fall

Der elastische Stoß wird in vielen Lehrbüchern behandelt. Eine gute Beschreibung des Problems findet man bei Landau [LAN81], so daß hier nur das Ergebnis gezeigt werden soll.

$$\frac{v_1'}{v} = \frac{m_1}{m_1 + m_2} \cos\Theta \pm \frac{1}{m_1 + m_2} \sqrt{m_2^2 - m_1^2 \sin^2\Theta}$$

(Bei $m_1 > m_2$ sind beide Vorzeichen vor der Wurzel zulässig, bei $m_2 > m_1$ nur das positive).
v_1' ist die Geschwindigkeit des Teilchens 1 nach dem Stoß, und v ist die Geschwindigkeit dieses Teilchens vor dem Stoß.
Damit ergibt sich für

$$E(\Theta) = E_0 * \left(\frac{v_1'}{v} \right)^2 ,$$

wenn E_0 die kinetische Energie des Teilchens 1 vor dem Stoß ist.

C.2 Der relativistische Fall

Im Schwerpunktsystem verändern sich die Impulsbeträge der Teilchen 1 und 2 durch den Stoß nicht. Es ändert sich lediglich die Impulsrichtung.
Ohne Einschränkung der Allgemeinheit werden zwei Annahmen gemacht:

* $\vec{P}_1$ vor dem Stoß sei in Richtung der X-Achse und
* die Z-Komponente von $\vec{P}_1$ nach dem Stoß sei identisch Null.

Nun werden Energie und Impuls des Teilchens 1 in bekannter Weise in das Laborsystem transformiert. Die Größen werden im Laborsystem mit einem Strich gekennzeichnet. Wie üblich wird die Lichtgeschwindigkeit identisch 1 gesetzt.

$$
\begin{aligned}
c &= 1 \\
p'_{1x} &= \gamma^* \ (p_{1x} - \beta^* E_1) \\
p'_{1y} &= p_{1y} \\
p'_{1z} &= p_{1z} \\
E'_1 &= \gamma^* \ (E_1 - \beta^* p_{1x}) \\
\beta^* &= - \frac{|\vec{P}_2|}{E_2} \qquad \text{und} \qquad \gamma^* = (1 - \beta^{+})^{-\frac{1}{2}}
\end{aligned}
$$

β^* ist die Geschwindigkeit des Schwerpunktsystems relativ zum Laborsystem. Das Laborsystem bewegt sich mit dem Teilchen 2 antiparallel zu der X-Achse. Da $p_z = 0$ ist, wird

$$\tan \Theta = \frac{p'_y}{p'_x} \qquad \text{Gl. (C.1)}$$

Mit Gl. (C.1) und der Transformation findet man nach einigen Umformungen einen Ausdruck für p_{1x}.

$$p_{1x} = \frac{b}{a} \pm \sqrt{ \left(\frac{d}{a} - \frac{c}{a} \right) + \left(\frac{b}{a} \right)^2 } \qquad \text{Gl. (C.2)}$$

mit

$$
\begin{aligned}
a &= \gamma^{*2} + \frac{1}{\tan^2 \Theta} \\
b &= \beta^* \gamma^{*2} E_1 \\
c &= \beta^{*2} \gamma^{*2} E_1^2 \\
d &= p_1^2 \ \frac{1}{\tan^2 \Theta}
\end{aligned}
$$

Aus p_{1x} und p_1 kann p_{1y} bestimmt werden. Durch Transformation erhält man die X- und Y-Komponente des Impulses im Laborsystem und daraus die gesuchten Betrag von E'_{1post}, der Energie des Teilchens 1 **nach** dem Stoß.

Zuvor müssen noch E_1 und p_1 bestimmt werden. Aus der Invarianz des Vierer-Impulses folgt

$$\left(E_1' + E_2' \right)^2 - p_1'^2 = \left(E_1 + E_2 \right)^2 - 0 \quad . \qquad \text{Gl. (C.3)}$$

E_1' und p_1' sind Energie und Impuls des Teilchens 1 **vor** dem Stoß. Nach Umformungen erhält man

$$p_1'^2 \; \frac{1}{1 + \left(\frac{m_1}{m_2} \right)^2 + 2 \frac{E_1'}{m_2}} = p_1^2 \quad .$$

E_1' und p_1' sind durch die kinetische Energie und der Ruhemasse m_1 vorgegeben.

$$E_1' = E_{1kin}' + m_1 \qquad \text{und} \qquad p_1'^2 = E_1'^2 - m_1^2$$

Literaturverzeichnis

[BOR85] M. Bormann, E. Fretwurst, G. Lindström, U. Pein, H.-Chr. Schleyer
Test measurements with a silicon-lead sandwich calorimeter for electromatic showers at energies between 1.5 and 5 GeV
NIM, A240 (1985) 63-68

[BOR87] M. Bormann, E. Fretwurst, G. Lindström, U. Pein, V. Riech, H-Chr. Schleyer, H. Weiss
Beam tests with large area silicon sandwich calorimeters for electron energies between 1 and 6 GeV
NIM, A257 (1987) 479-487

[FLA39] A. Flammersfeld
Die untere Grenze des kontinuierlichen β-Spektrums des RA E.
Zeitschrift für Physik, Vol. 112 (1939) 727-743

[FUN89] Herr M. Funk, mündliche Mitteillung
I. Institut für Experimentalphysik in Hamburg, April 1989

[JOST46] R. Jost
Bemerkungen zur mathematischen Theorie der Zähler
Helvetia Physica Acta, Vol. 20 (1946) 173-182

[KDJ84] Digital Equipment Corporation
KDJ11-A, CPU Module, User's Guide
EK-KDJ1A-UG-001 (1984)

[KITT83] C. Kittel
Einführung in die Festkörperphysik
R. Oldenbourg Verlag München Wien (1983)

[LAN81] L. D. Landau, E. M. Lifschitz
Lehrbuch der Theoretischen Physik I, Mechanik
Akademie-Verlag, Berlin (1981)

[LIF38] H. Lifschutz, O. S. Duffendack
The counting losses in Geiger-Müller counter circuits and recorders
Physical Review, Vol. 54 (1983) 714-725

[LON48] K. Lonsdale
Geiger counter mearsurements of Bragg and diffuse scattering of X-rays by single crystals
Acta Crystallographica, Vol. 1 (1948) 12-20

[PET62] W. Petzold
Messungen von Zählverlusten durch Vergleich zweier unabhängiger Zählkreise.
Zeitschrift für angewandte Physik, Vol. 15 (1963) 158-160

[PUS83] A. G. Puskeppel, M. Niecke
Phalst - System für Rechnergestützte PIXE-Mikrostrahl-Messungen
Jahresbericht des Instituts (1982/83) 162-164

[RUA37] A. E. Ruark, F. E. Brammer
The efficiency of counters and counter circuits
Physical Review, Vol. 52, 322-324 (1937)

[RUT77] Ch. Rutkowski, G. Lindström, V. Riech
Messungen zur Energiebreite des analysierten Strahls am Hamburger Isochronzyklotron
Jahresbericht des Instituts (1976/77) 132-133

[SIC89] E. Borchi, R. Macii, S. Mazzoni, I. Fedder, G. Lindström, C. Bertrand, F. Lamarche, C. Leroy, A. Villari, M. Bruzzi, C. Furetta, R. Paludetto, S. Pensotti, P. G. Rancoita, C. Simeone, L. Venturelli, L. Vismara, J. E. Brau, N. Croituro, A. Seidman, S. C. Berridge, W. M. Bugg, R. Giacomich, A. Penzo, E. Toppano, P. Giubellino, L. Ramello, L. Riccati, M. Pisani, R. Steni
Silicon sampling hadronic calorimetry: A tool for experiments at the next generation of colliders
International conference on advanced technology and particle Physics,
CERN-EP/89-28, Feb. 1989

[TPR87] E. Fretwurst, G. Lindström, V. Riech
Technical progress report for the development of the PLUG calorimeter
Sep. 87, not published

[VOG85] W. Vogel
Ein neues Multiparameter-Messdaten-Verarbeitungssystem und ausgewählte Anwendungsbeispiele
Dissertation am Fachbereich Physik der Universität Hamburg, 1985

[WILL62] C. Williamson, J. P. Boujot
Tables of range and rate of energy loss of charged particles
Centre d' Etudes Nucleaires de Saclay, 1962

[WIR78] N. Wirth
Systematisches Programmieren
Teubner Studienbücher, Informatik, 1978

Danksagung

An dieser Stelle möchte ich allen, die zum Gelingen dieser Arbeit beitrugen, meinen herzlichen Dank aussprechen. Mein besonderer Dank gilt:

* Herrn Prof. Dr. G. Lindström für die Aufgabenstellung und die angenehme Betreuung.

* Herrn Dr. E. Fretwurst für seine Präsenz in wichtigen Fragen.

* Herrn I. Fedder für seine Mitwirkung am Testexperiment und für seine gründliche und konstruktive Kritik.

* Herrn A. Puskeppel für die Unterstützung und Beratung in vielen Hardware- und Software-Fragen.

* Herrn U. Pein für seine Mitarbeit beim Versuchsaufbau am Isochronzyklotron.

* Herrn Nürnberg für seinen besonderen Einsatz während der Meßzeiten am Isochronzyklotron.

* Allen MitgliederInnen der "Nuklearen Meßtechnik" für das freundliche Arbeitsklima.

* *Last but not least*, allen "Mitstreitern" für die Unterstützung beim Testexperiment und für die vielen Diskussionen vor allem fachübergreifender Themen.

Erklärung

Hiermit erkläre ich, die Arbeit selbstständig und unter Angabe der wesentlichen Hilfsmittel verfaßt zu haben.

Hamburg, den 2. Mai 1989