TECHNISCHE
UNIVERSITÄT
DARMSTADT

INSTITUT FÜR KERNPHYSIK

DIPLOMARBEIT

von

Andreas Stiller

Entwicklung der Ausleseelektronik für ein Vieldraht-Driftkammer System am Darmstädter Q-Clam Spektrometer

Februar 1991

Inhaltsverzeichnis

1 Einleitung

Am Institut für Kernphysik der TH Darmstadt werden zur Zeit die Einrichtungen für Elektronenstreuxperimente bis zu dem Energiebereich von 130 MeV aufgebaut. Es sind sowohl (e,e') als auch (e,e'x) Experimente geplant. Dabei bedeutet e'x den koinzidenten Nachweis von Elektron und $x = p, .., \alpha, n, \gamma$ oder Spaltprodukten.

Kernstück der Anlagen ist der neue supraleitende 130 MeV Elektronenbeschleuniger S-DALINAC, der einen cw-Strahl (continous wave) mit einer Intensität von 20 μA und einer relativen Energieunschärfe von 10^{-4} bei 130 MeV liefern wird [1]. Alle geplanten Experimente benötigen einen impulsauflösenden Elektronendetektor. Der neue Meßplatz benutzt dazu ein Magnetspektrometer vom Typ Q-Clam und Vieldraht-Driftkammern als orts- und winkelauflösenden Detektor.

Die Elektronenstreuwirkungsquerschnitte sind vor allem bei Koinzidenzexperimenten sehr niedrig. Um akzeptable Meßzeiten zu erhalten, muß der Raumwinkel des Spektrometers möglichst groß sein. Die Koinzidenzexperimente werden bei Anregungsenergien oberhalb der Teilchenschwelle und unterhalb des quasielastischen Bereichs, im Riesenresonanzbereich, durchgeführt. Es sollte deswegen möglich sein, einen großen Anregungsenergiebereich zu überdecken. Daher hat das neue Spektrometer einen Raumwinkel von 40 mSr und eine Impulsakzeptanz $\frac{\Delta p}{p}$ von 20 %.

Diese günstigen Eigenschaften verlangen erheblichen Aufwand bei der Auswertung, da

- wegen der hohen Winkelakzeptanz ($\pm$ 100 mr) der Streuwinkel einmal zur Festlegung des Korrelationswinkels, zum anderen für die Rückstoßkorrektur berechnet werden muß.

- die Fokalebene stark gekrümmt ist und damit die Energiebestimmung allein mit Hilfe der Dispersion nicht möglich ist.

Die damit notwendige numerische Umkehrung der Spektrometerabbildung ist zeitaufwendig und verzögert die physikalische Auswertung. Da die notwendigen Algorithmen aber detektorspezifisch sind, besteht die Möglichkeit, sie von der Ausleseelektronik der Driftkammern während der Messung ausführen zu lassen.

Das Driftkammer-System ist eine Neuentwicklung [2] unter Berücksichtigung von Erfahrungen mit einer Driftkammer an dem 120° Spektrometer [3] am DALINAC, einem gepulsten 70 MeV Elektronenbeschleuniger [4].In dieser Entwicklung wurde sehr früh die Schnittstelle zwischen detektorspezifischer Elektronik und nachfolgender Datenerfassung definiert. So bestand die Möglichkeit zur zeitlich parallelen Entwicklung einer Ausleselektronik, die neben dem Anschluß an das übergeordnete Datenerfassungssystem auch genügend Rechenleistung zur sofortigen (Online) Analyse bietet.

Ziel der vorliegenden Arbeit war es, eine Ausleseelektronik für die neuen Vieldraht-Driftkammern zu entwickeln, die mit der Nutzung von Mikroprozessoren folgende Aufgaben erfüllen soll:

- **Erfassung und Transport der Detektor-Rohdaten.**
  Dazu gehört die Überwindung der Entfernung zwischen Driftkammerelektronik und Datenerfassung von etwa 50m unter Beachtung vorgegebener Schnittstellen. Dabei soll galvanische Trennung der beiden Systeme erreicht werden.

- **Kalibrierung des Detektors**
  Vor der Erzeugung physikalisch relevanter Spektren müssen die Rohdaten (Driftzeiten etc.) des Spektrometer/Driftkammer Detektors in den Elektronenimpuls (Energie, Streuwinkel) transformiert werden. Diese Konvertierung ist detektorspezifisch und wird für jeden Datensatz (Event), der ein Streuereignis beschreibt, ausgeführt.

  Die beteiligten Algorithmen bleiben nach ihrem Test für jedes Experiment gleich. Dieser Teil der Auswertung kann daher als Kalibration des Detektors verstanden werden. Ihre Ausführung soll der Ausleseelektronik übertragen werden können, so daß der Detektor neben seinen Rohdaten den Elektronenimpulsvektor liefert.

- **Geringe Totzeit**
  Die Rechenleistung soll so hoch sein, daß bei einem (e,e'x) Experiment mit Zählraten in der Größenordnung von 10 - 1000 Hz nur geringe Totzeitverluste entstehen.

- **Universelle Einsetzbarkeit** des Mikroprozessoranteils
  Die Mikroprozessor (μP) Module als wesentlicher Bestandteil der Ausleseelektronik sollen auch einsetzbar sein für Aufgaben in der Beschleunigersteuerung. Das betrifft die Hardware, die Entwicklungsumgebung, Bibliotheksfunktionen und Gerätetreiber, die im Rahmen dieser Arbeit angepaßt bzw. entwickelt werden sollen.

2 Grundlagen und Schnittstellen

Die Ausleseelektronik ist das Bindeglied zwischen den Analog-Digitalkonvertern des Driftkammer Systems [2] und der Datenerfassung des Experiments. [5]. Beide Komponenten sollen im folgenden erläutert werden mit dem Ziel, die Schnittstellen für die Ausleseelektronik offenzulegen.

2.1 Vieldraht-Driftkammer

Ein einfacher Gaszähler zum Nachweis geladener Teilchen ist das Proportionalzählrohr. Wird eine Anordnung von mehreren Zählrohren gewählt, in der die Zähldrähte parallel und mit festem Abstand zueinander in der Nachweisebene liegen, so ist der Prinzipaufbau und zugleich die einfachste Betriebsart einer Vieldraht-Driftkammer verwirklicht (Abbildung 2.1). Dabei befinden sich die Zähldrähte in einem gemeinsamen Gasvolumen und sind gegen Übersprechen zwischen den einzelenen Zellen geschützt durch abwechselend eingefügte, auf Massepotential liegende inaktive Drähte. Die elektrischem Feldlinien, die zu verschiedenen Zähldrähten führen, sind dadurch getrennt und definieren eine Driftzelle.

Die Ortsauflösung von einem Drahtabstand wird wesentlich gesteigert im Betrieb als vertikale Driftkammer. Dabei wird innerhalb jeder Driftzelle, in der das Teilchen nachgewiesen wurde, sein mimimaler Abstand zur Drahtebene ermittelt (Abbildung 2.2). Durch geeignete Wahl des Winkels zwischen Nachweisebene und Extremstrahlen der Teilchen wird sichergestellt, daß genügend Driftzellen zur Beschreibung einer Teilchengeraden ansprechen. Die Gerade liefert mit ihrem Schnittpunkt durch die Drahtebene und ihrer Steigung eine Orts- und Winkelauflösung, die von der Zahl und der Genauigkeit der gemessenen Abstände abhängt.

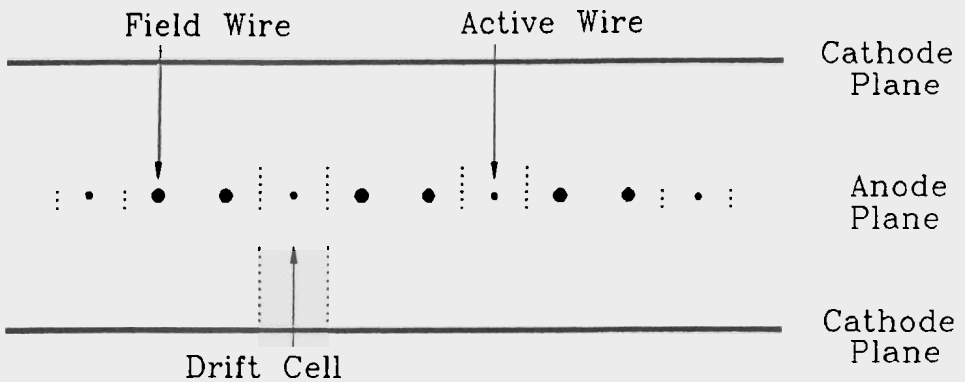

Abb. 2.1: Schema des Aufbaus einer Vieldraht-Driftkammer. Die Driftzellen sind mit Hilfslinien gekennzeichnet.

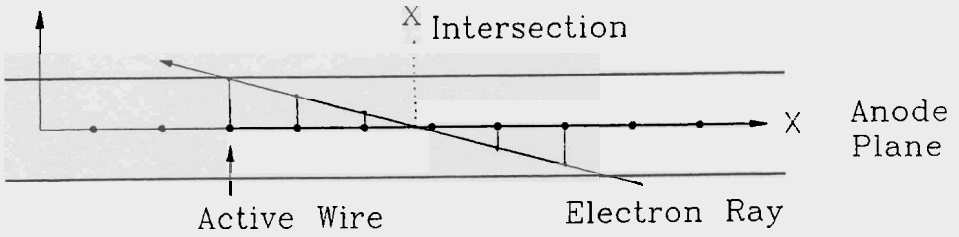

Abb. 2.2: Erhöhung der Ortsauflösung durch Abstandsmessungen innerhalb jeder Driftzelle

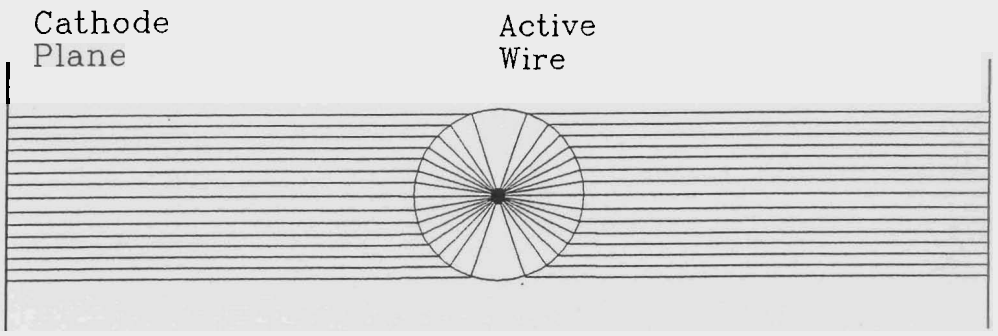

Abb. 2.3: Vereinfachte Form der elektrischen Feldlinien innerhalb einer Driftzelle. Die Übergangszone zwischen den beiden Feldregionen ist durch den Kreis gekennzeichnet. Den exakten Feldverlauf findet man in [6].

Das Meßprinzip zur Abstandsbestimmung beruht auf folgenden Grundlagen [6]:

- Das elektrische Feld innerhalb der Driftkammer ist periodisch in den einzelnen Driftzellen. Innerhalb einer Driftzelle teilt es sich in zwei Regionen (Abbildung 2.3). Der erste Teil von der Kathodenfolie bis in den Nahbereich der Drähte ist gleichförmig und im Proportionalbetrieb so schwach, daß keine Gasverstärkung auftritt. Im Nahbereich der Drähte herrscht dagegen ein radiales, mit $\frac{1}{r}$ ansteigendes Feld, das Sekundärionisation und damit Gasverstärkung verursacht.

  Der Drahtabstand bestimmt den Durchmesser des radialen Feldbereichs. Daher kann durch eine Vergrößerung des Abstandes zwischen Kathodenfolie und Drahtebene erreicht werden, daß der größte Weganteil der erzeugten Ladungsträger durch ein gleichförmiges Feld verläuft.

- Die Geschwindigkeit der Ladungsträger ist nach wenigen freien Weglängen im homogen, schwachen Feldbereich nahezu unabhängig von Gasmischung und Druck konstant. Die Ladungsträger driften entlang der Feldlinien in den Nahbereich des Drahtes hinein.

Die Zeit zwischen der Primärionisation und dem dem Nachweis der Ladungsverschiebung an den Zähldrähten ist daher proportional zu dem gesuchten Abstand zwischen

dem Schwerpunkt der Primärionisation und der Nachweisebene. Dieser Zusammenhang wird durch Korrekturen ergänzt, die den radialen Feldverlauf in Drahtnähe berücksichtigen. Mit einem schnellen Detektor (Szintillator) parallel zur Nachweisebene, dessen Zeitinformation möglichst ortsunabhängig ist, läßt sich der Zeitpunkt der Primärionisation definieren.

Der gesuchte Abstand kann also aus der gemessenen Zeit zwischen den Signalen von Triggerdetektor und Zähldraht ermittelt werden. Diese Meßgröße enthält aber keine Information darüber, ob die Ionisation über oder unter der Drahtebene erfolgte. Diese Mehrdeutigkeit muß bei der Rekonstruktion der Teilchengeraden berücksichtigt werden.

Die besonderen Eigenschaften des Spektrometers, also stark gekrümmte Fokalebene, großer Raumwinkel und große Impulsakzeptanz, erfordern zur Impulsberechnung die Kenntnis des gesamten Elektronenvektors nach der Abbildung durch das Spektrometer. Die geforderten Genauigkeiten in den Komponenten hängen von Vergrößerungen durch das Spektrometer ab. Die Komponenten sind:

x - Dispersionsrichtung in der Nachweisebene
y - senkrecht dazu in der Nachweisebene
θ - abgebildeter Streuwinkel
ϕ - abgebildeter Azimutalwinkel

Die Bestimmung dieser Größen erfolgt für dieses Spektrometer mit drei Driftkammern X1,Y und X2. Die X-Kammern weisen in Dispersionrichtung nach, die Y-Kammer um 30° gegen die Dispersionsrichtung geneigt. Die beiden X-Kammern erhöhen die Winkelauflösung für ϕ, während θ nur mit der Y-Kammer bestimmt wird.

Die wichtigsten Paramter der neu entwickelten Driftkammern sind in Tabelle 2.1 zusammengefaßt.

Tabelle 2.1: Geometrie und Betriebsparamter der Vieldraht-Driftkammern

Größe	Wert
Zahl der aktiven Drähte	X1,Y: 128 und X2: 160
Nachweisfläche	1000 mm * 120 mm
Drahtdurchmesser	**aktiver Draht** : 20 μm, inaktiver Draht: 50 μm
Wicklungsperiode	aktiver Draht und zwei inaktive Drähte
Drahtabstand	2 mm, zwischen aktiven Drähten 6 mm
Abstand Drahtebene zu Kathodenfolie	12 mm
eingesetzte TDCs	**acht** Zähler mit 1 Ghz Taktfrequenz
Zählgas	Argon - Isobutan Gemisch

Das geschilderte Meßprinzip erfordert folgende Konverter für **jeden** Nachweisdraht:

- Der **Nachweis** der Drahtsignale erfolgt mit Vorverstärker, Impulshöhen-Diskriminator und Hitpattern-Register. Die Impulshöhen sind im Betrieb als vertikale Driftkammer keine Meßgröße. Nur die Information, ob und wann eine Driftzelle vom Elektron durchdrungen wurde, wird weiterverarbeitet.
- Die Messung der **Driftzeiten** erfolgt mit Time to Digital Convertern (TDC's). Der Wertebereich hängt ab von von der Driftgeschwindigkeit im Zählgas und der Länge der Driftzelle und liegt hier zwischen 0 und 220nsec.

Zwei Abweichungen von dieser idealisierten Anordnung sind bei dem neuen Driftkammer-System zu nennen:

1. Die Zahl der notwendigen TDC's wurde verringert. Dadurch wird der Platzbedarf geringer und durch die Reduzierung der Modulanzahl die Wartung der Elektronik erleichtert.

   Die Grundlage für diese Vereinfachung ist:

   - Durch die Wahl des Drahtabstandes und der Festlegung des Winkels zwischen Drahtebene und den Randstrahlen der Elektronen kann die maximale Zahl n von Drähten festgelegt werden, die pro Elektron ansprechen. Mehrfachereignisse innerhalb des Nachweisfensters des Detektors (hier $\sim$ 500 nsec) treten im Betrieb mit einem cw-Strahl nur mit geringer Wahrscheinlichkeit auf. Ihr Nachweis ist schwierig, da die Zeitpunkte der Primärionisationen nur mit einem segmentierten Szintillator nachzuweisen wären. In diesem System werden Mehrfachereignisse verworfen.
   - Die Zähldrähte lassen sich in N Gruppen aufteilen, in denen die Drähte einen Zählabstand von N haben:

     1.Gruppe: 0,N,2N,...
     2.Gruppe: 1,1+N,1+2N,...
     N.Gruppe: N-1,2N-1,3N-1,...

     Das logische Oder der Nachweisdetektoren innerhalb einer Gruppe liefert das Zeitsignal für die N TDC's. Die eindeutige Zuordnung von TDC zu Zähldraht ist bei Ausschluß von Mehrfachereignissen für $N >= n$ gewährleistet.

2. Das Szintillator-Signal wird durch seine Wandlerelektronik (Photomultiplier und Mean-Timer) so weit verzögert, daß für drahtnahe Ereignisse das Startsignal nach dem Stop des Zähldrahtes eintreffen kann. Daher müßten alle Drahtsignale mit Delay-Lines verzögert werden.

$$\begin{aligned}\Delta t &= (t_0 + t_{Drift} + t_{Drift-Delay}) - (t_0 + t_{Szintillator-Delay}) \\ &= (t_{Drift-Delay} - t_{Szintillator-Delay}) + t_{Drift} \\ &\quad \text{mit } t_{Drift-Delay} >= t_{Szintillator-Delay} \\ &\quad \text{und } t_0 = \text{Zeitpunkt der Primärionisation}\end{aligned}$$

Vertauscht man allerdings Start und Stop der TDC's, so werden Kabelverzögerungen der Drahtsignale nicht benötigt. In dieser Anordnung dient das Szintillatorsignal als Stop und die unverzögerten Drahtsignale als Start. Damit werden allerdings gespiegelte Driftzeiten gemessen:

$$\begin{aligned} \Delta t &= (t_0 + t_{Delay}) - (t_0 + t_{Drift}) \\ &= t_{Delay} - t_{Drift} \\ &\quad \text{mit } t_{Delay} >= max(t_{Drift}) \end{aligned}$$

Die maximale Driftzeit muß zur Bestimmung des Zeitnullpunktes für ein eingestelltes Szintillator-Delay gemessen werden.

Die Schnittstelle zur Ausleseelektronik ist weiterhin durch folgenden Vorgaben bestimmt:

- Die Elektronik ist für jede Kammer seperat aufgebaut. Die Platinen sind in VME - Crates montiert.

- Alle Wandler sind über einen Bussystem anzusprechen. Die zu bedienenden Wandler sind in Tabelle 2.2 zusammengerfaßt.

Tabelle 2.2: Digitalkonverter der Driftkammer Elektronik

Typ	Zugriff	Anzahl	Kodierung
Hitpattern	Byte-Register	16 od. 20	1/0 $\leftrightarrow$ no hit/hit
Erster angesprochener Draht	Byte-Register	1	invertiert
Status, ob Hitpattern vollständig ausgelesen wurde	Bussignal		
TDC	Wort-Register	8	12 Bit, digit $\sim$ 1 nsec
Diskriminator-Schwelle	Byte-Register	16 od. 20	digit $\sim$ 1 mV

2.2 Datenerfassung

Zur Aufnahme und Auswertung aller kernphysikalischen Experimente am neuen Meßplatz wird ein Programmpaket der Gesellschaft für Schwerionenforschung GSI verwendet. Das GSI-Online-Offline- System GOOSY bietet mit Kommandozeilen oder Menues als Benutzerschnittstelle und PL/I-Programmbibliotheken die in Tabelle 2.3 gezeigten Komponenten.

Tabelle 2.3: GOOSY - Komponenten

Modul	typische Anwendungen
Daten-Strukturierung	Definition von Eventstruktur, Spektren, Kalibrierungen, ...
Analyse	Akkumulation von Spektren, prüfen von Polygon-Conditions in zwei-dimensionalen Spektren, ...
Display	Anzeige/Druck von Spektren, interaktives Erzeugen von Polygon-Conditions, Online-Darstellung von Scatterplots, ...
Aufnahme	Einlesen der Experimentdaten über Geräte-Treiber des Betriebssystems, Weiterleiten der Daten zur Analyse (auch zu anderen Rechnern), ...

Das GOOSY wurde auf den Institutsrechnern, einem VAX-Cluster aus VAX-3800, VAX-3600 und mehreren Workstations, unter VAX/VMS installiert [5]. GOOSY erfaßt die Meßdaten im Listmode, alle Spektren werden auschließlich von GOOSY akkumuliert, die Meßdaten eines Streuereignisses (Event) sind die kleinste Dateneinheit.

Da der Verwaltungsaufwand des Betriebsystems für eine I/O-Operation relativ hoch, (0.5 - 10 msec) aber weitgehend unabhängig von der Länge des Datensatzes ist, wurde als Schnittstelle zur Experimentelektronik der Event-Buffer gewählt. In dem Event-Buffer werden von der Experimentelektronik mehrere Events zwischengespeichert, die dann mit einer I/O Operation zum Datenerfassungsrechner transferiert werden können.

Ein Interface zur Übertragung der Event-Buffer bildet mit seinem Treiber-Programm den Übergang zwischen GOOSY und der Experimentelektronik. Da die Experimentelektronik für die Bildung des Event-Buffers verantwortlich ist, muß sie zwangsläufig einen Mikroprozessor enthalten, der das Auslesen und Zwischenspeichern der Events übernimmt.

Im vorliegenden Fall wird CAMAC [7] als Modulnorm in der Multi Branch Konfiguration gewählt. Ein CAMAC Branch verbindet das System Crate, das durch sein Interface zum Rechnersystem ausgezeichnet ist, mit bis zu sieben anderen CAMAC Überrrahmen. Innerhalb eines Branches können bis zu sieben Crates adressiert werden.

Das neu installierte GOOSY benutzt ein System Crate Interface SCI 2280/81 der Firma C.E.S und einen speziellen VMS-Treiber [5]. Zur Unterstützung der GOOSY-Datenerfassung dienen folgende Mikroprozessoren (Frontendprozessoren) innerhalb des CAMAC Systems:

- Jedes Crate enthält einen eigenen Mikroprozessor (Auxilliary Crate Controller ACC 2180 der Firma C.E.S), der alle Meßdaten innerhalb seines Crates ausliest uns zu einem Subevent zusammenstellt.

- Das System-Crate enthält ebenfalls einen ACC 2180. Dieser kann aber über alle Branches auf alle Crates zugreifen. Dieser Mikroprozessor ist der übergeordnete Rechner (Event-Builder), der auf alle Subevents wartet und sie zu dem eigentlichen Event zusammenfügt. Er verwaltet den Event-Buffer und veranlaßt die Ausgabe-Operation an die VAX über das System Crate Interface.

Auf Grund dieser flexiblen Erfassung der Meßdaten, ist die Schnittstelle zur Driftkammer-Ausleseelektronik nur von der CAMAC Norm bestimmt. Das Datenformat ist frei wählbar, es spiegelt sich in den Ausleseprogrammen der ACCs wieder.

3 Entwurfsentscheidungen

Dieses Kapitel soll die Grenze zwischen einleitenden Grundlagen und der Entwicklungsarbeit innerhalb dieser Arbeit markieren. Der Übergang wird geliefert durch einige Entscheidungen, die die gesamte Entwicklung bestimmt haben.

3.1 Implementation als Embedded System

Die Schnittstellendefinition zur GOOSY Datenerfassung legt nahe, die gesamte Ausleselektronik für die Vieldraht-Driftkammern (Multiple Wire Drift Chamber MWDC) als für den Benutzer unsichtbaren Teil eines CAMAC-Moduls aufzufassen. Nach Einstellung der Geräteparameter hat hat es als einzige Aufgabe, Meßdaten zu liefern. Das schließt Programmentwicklung, Multiuser-Betrieb oder andere Aufgaben, die nicht der GOOSY-Kontrolle unterliegen, auf den Rechnern des MWDC Readouts aus.

Diese Reduktion der Möglichkeiten auf den Meßbetrieb spart die Investition für ein eigenes Betriebssystem. Sein Wegfallen erübrigt neben eigenen Massenspeichern und Terminals auch die Hardware zur Umsetzung eines virtuellen Speichermodells, das in moderne Betriebsystemem allen Benutzern einen eigenen, geschützten Adressraum zur Verfügung stellt.

Ein Mikroprozzessor-Modul ohne Betriebssystem, das sich auf seine Hauptaufgabe beschränkt, und das in ein Gerät, möglichst für den Benutzer unsichtbar, eingebettet ist, wird auch als 'embedded system' bezeichnet.

3.2 Cross-Entwicklung auf den Institutsrechnern

Die Plazierung aller Entwicklungswerkzeuge auf den Institutsrechnern hat neben der kostengünstigen Nutzung ihrer Betriebsmittel (Rechenzeit, Massenspeicher) weitere Vorteile:

- Die Entwicklungswerkzeuge sind zwangsläufig in die bekannten VMS-Benutzerschnittstellen, DEC Command Language (DCL) und DecWindows eingebunden.
- Vorhandene Hilfsmittel zur Unterstützung der VAX-Software Entwicklung können mitbenutzt werden.
- Das VMS-Filesystem mit der VAX-Cluster Software bietet Zugang zu allen Massenspeichern unabhängig von dem Rechner, an dem sie angeschlossen sind. Die Entwicklung ist damit von allen Arbeitsplätzen, Terminals und Workstations, des Instituts aus möglich.

3.3 Prozessorfamilie Motorola MC68020

Die Leistungsdichte im Angebot an Mikroprozessoren der 32-Bit Generation erlaubte die Auswahl auf Grund früherer Erfahrungen [8, 9] und Entscheidungen andererer Institute (CERN, GSI). Die gewählte Prozessorfamilie MC68020 der Firma Motorola [10, 11] hat folgende Merkmale:

- Complex Instruction Set Computer (CISC)
  Allen grundlegende Datentypen bis zur Größe von 32 Bit werden mit 16 Universalregistern und komplexen Adressierungsarten unterstützt. Die Fließkomma Operationen und Datentypen werden mit einem Koprozessor MC68881 implementiert.

- Linearer Adressraum im physikalischen Speichermodell mit der Größe von 2^{32} Bytes. Das virtuelle Speichermodell benötigt eine externe Memory Managment Unit MMU MC68851.

- Memory mapped I/O
  Der Anschluß von Peripheriegeräten hat die gleiche Busschnittstelle und benutzt den gleichen Befehlssatz wie für Speicherzugriffe. Die Interruptstruktur hat acht Ebenen mit 256 Vektoren.

- Dynamic Bus Sizing
  Die Datenbusbreite eines Speicherbereiches oder eines Peripheriegerätes ist durch Quittungssignale während des Zugriffes festzulegen. Mögliche Werte sind 8,16,24 und 32 Bit. Zugriffe auf Operanden länger als die quittierte Portbreite oder mit ungeraden Anfangsadressen werden automatisch in mehrere Zyklen zerlegt.

- Multiprozessorfähig
  Alle Datentypen bis zu einer Größe von 2 * 32 Bit (z.B. Zähler, Queues, Semaphoren) können auch bei gleichzeitigen Zugriff mehrerer Prozessoren mit einem ununterbrechbaren Speicherzugriff (Read Modify Write cycle) und den zugeordneten Befehlen verändert werden.

  Das ist entscheidend für die korrekte Implementation einer Interprozessor Kommunikation. Im Fehlerfall könnte eine Speicherzelle gelesen, vom Prozessor verändert und zurückgeschrieben werden, während ein zweiter Rechner genau in dieser Zeitspanne ebenfalls in die Datenstruktur eingreift. Einer der beiden Prozessoren ginge dann zwangsläufig von veralteten Informationen aus, wenn er die Datenstruktur verändert.

3.4 Multiprozessor-System mit variabler Topologie

Das Darmstädter MWDC Design setzt drei Driftkammern ein. Ihre Digititalkonverter sind getrennt in drei Crates aufgebaut. Auch Teile der geplanten Online-Kalibrierung, die Bestimmung der Durchstoßpunkte durch die Drahtebenen, sind unabhängig voneinander. Durch die Zuordnung eines Prozessors zu jeder Kammer und der Zusammenführung ihrer Zwischenergebnisse in einem vierten Prozessor, der die weiteren

Berechnungen übernimmt, sollte die Trennung der Kammern zu einer Reduzierung der Totzeit genutzt werden.

Diese Idee führte zu der weitergehenden Entscheidung, ein Multiprozessor-System zu entwickeln, dessen Topologie variabel ist. Damit sollten auch andere Aufteilungen des Rechenflusses auf ein Multiprozessor System abgebildet werden können.

Diese Forderung bedingt ein Bussystem zur Rechnerkopplung, bei dem jeder Rechner mit allen anderen verbunden ist. Die VME Spezifikation [12] als Standard für den Systembus der Motorola Prozessoren wurde hierfür gewählt. Seine Definition gibt zwar keine spezielle Vorschriften (z.B. Mailboxen oder Interprozessor Interrupts) für die Kommunikation mehrerer Prozessoren. Sie erlaubt aber mehrere aktive Benutzer des Systembusses und beschreibt die zu verwendende Zuteilungslogik.

Es wurde daher die Lösung mit dem geringsten Hardware-Aufwand gewählt. Jeder Prozessor bietet dabei seinen lokalen Arbeitsspeicher auch über den VME Bus an. Damit können beliebige Datentypen zur Kommunikation zwischen den Prozessoren verwendet werden. Speicherzugriffe, die in einen anderen Prozessorspeicher führen, müssen die Buszuteilung abwarten und dann den Zielprozessor suspendieren, um den Zugriff auf seine Arbeitsspeicher zu ermöglichen. Der relativ hohe Zeitaufwand dieser Operationen wird innerhalb des MWDC-Systems in Kauf genommen, da die Berechnungen während der Ausführung der Kalibrationsalgorithmen gegenüber dem reinen Datentransport überwiegen und die Lokalität der Daten im Speicher der auswertenden Prozessoren während der Berechnung garantiert werden kann. Alle Datenstrukturen zur Kommunikation, auf die konkurrierend zugegriffen werden kann, müssen mit einem einzigen ununterbrechbaren Zugriff verändert werden können.

3.5 Multitasking zur Problemzerlegung

Multitasking ist ein Hilfsmittel zur Entwicklung von Programmen. Dabei wird versucht das Gesamtproblem in mehrer übersichtliche Einheiten (Tasks) zu zerlegen. Die Kommunikation und Synchronisation der Teile regelt ein Multitasking Kern. Das ist ein Programmpaket, das die Implementation des Multitasking von dem Benutzter abgrenzt. Die Verwaltung der notwendigen Datenstrukturen und die Zuteilung der Rechenzeit wird von dem Kern übernommen, der seine Befehle von den Tasks und Interrupt Service Routinen über Unterprogrammaufrufe erhält.

Der Einsatz des Multitasking in allen Aufgaben für die zu entwickelten Mikroprozessorsystem soll daher durch einen kommerziellen standardisierten Kern unterstützt werden.

Die im vorigen Kapitel angedeutete Aufteilung der Aufgaben des MWDC Programmpaketes, Transport und Transformation der Meßdaten, benutzt mehrerere Prozessoren als Teileinheiten. Das einzige Unterbrechungsignal (Interrupt), das den Zustand des Systems ändert, ist der Trigger der Meßelektronik.

Die Methoden des Multitasking: Aufteilung in mehrerer Tasks und deren Steuerung und Synchronisation durch Interrupts, Mailboxen, Signale, Timer und Gerätetreiber scheinen daher für diese Aufgabe überflüssig.

Die Möglichkeit des verwendeten Kerns (pSOS^{+m}[13]), alle Methoden des Multitasking auch über Prozessorgrenzen hinweg zu verwenden, begründeten seinen Einsatz auch im MWDC Readout. Initialisierung, Einstellung und Überwachung sollten damit von dem eigentlichen Meßprozess getrennt werden.

4 Struktur des MWDC Readouts

Die gesamte Struktur des MWDC Readouts wird bestimmt durch die Entscheidung zum Multiprozessorbetrieb. Ein Multiprozessorsystem erlaubt die gleichzeitige Bearbeitung mehrerer Probleme, während Multitasking die gleichzeitige Ausführung nur als Entwurfshilfe simuliert. Um den Durchsatz bei der Anwendung eines Algorithmus auf viele gleichförmige Datensätze (Events) zu steigern, bietet Multiprozessing drei Möglichkeiten:

- **Pipelines**
  Dabei hat der Algorithmus mehrere Teile, die nacheinander von spezieller Hardware ausgeführt werden müssen. Die Teileinheiten reichen reichen ihre Zwischenergebnisse weiter und stehen danach sofort für die Bearbeitung des nächsten Datensatzes bereit. Mit einem kontinuierlichen Datenstrom ist der Durchsatz nur durch das Maximum der einzelnen Ausführungszeiten begrenzt.

- **algorithmische Parallelisierung**
  Hier wird der Algorithmus so zerlegt, daß die Teile unabhängig von anderen Zwischenergebnissen ausgeführt werden können. Nach paralleler Berechnung auf mehreren Prozessoren müssen die Ergebnisse wieder zusammengeführt werden. Diese Methode ist bei komplizierten, Änderungen unterworfenen Algorithmen nur dann einzusetzen, wenn ein spezieller Compiler die Zerlegung und die Synchronisation übernimmt.

- **Verbreiterung des Datenstroms**
  Diese Form des Multiprozessing nutzt aus, daß die einzelnen Events voneinander unabhängig sind. Die eingesetzten Rechner bearbeiten mehrere Events parallel. Dabei muß nur auf die Erhaltung der Ergebnisreihenfolge geachtet werden. Spezielle Änderungen an der Kodierung des Algorithmus sind nicht nötig. Bei Vernachlässigung des Aufwandes für die Verteilung der Events auf die beteiligten Prozessoren, steigt der mittlere Durchsatz linear mit der Zahl der eingesetzten Prozessoren.

Die einzige Methode, die nicht nur den mittleren Durchsatz steigert, sondern auch die Ausführungszeit für eine Anwendung des Algorithmus senkt, ist die Parallelisierung innerhalb des Algorithmus. Die beiden anderen Methoden beruhen darauf, daß ständig mehrere Events vom System bearbeitet werden und können daher nur den Durchsatz steigern. Die Ergebnisse verlassen das System nach der gleichen Zeit wie bei der Benutzung eines Prozessors.

Es stand allerdings kein parallelisierender Compiler für den Prozessor MC 68020 zur Verfügung, so daß entschieden wurde, die Parallelisierung nicht einzusetzen innnerhalb der Kalibrationsalgorithmen. Die manuelle Zerlegung durch den Entwickler erfordert einen zu hohen zusätzlichen Aufwand auf Grund der zu erwartenden Änderungen der numerischen Verfahren innerhalb der Testphase. Auch die schlechten Testmöglichkeiten (mehrere Rechner sind an einem Event beteiligt) sprechen gegen diese Lösung.

Bei Koinzidenz-Experimenten muß allerdings die Zuordnung der Meßdaten des Elektronendetektors zu denen der anderen Detektoren für ein Streuereignis garantiert werden. Bei einfachem synchronen Auslesen aller Detektoren durch die GOOSY Frontendprozessoren bringen die beiden anderen Methoden daher keinen Vorteil. Sie steigern nur den Durchsatz und senken nicht die Ausführungszeit pro Event und damit würden die GOOSY Frontendprozessoren weiterhin auf die Ergebnisse nach jedem Trigger warten.

Eine Erweiterung im Programm des GOOSY Event-Builders [5] erlaubt dennoch den sinnvollen Einsatz von Pipelines und paralleler Bearbeitung mehrerer Events. Dabei wird genutzt, daß durch das Sammeln vieler Events vor der Übertragung zum GOOSY Rechner jedes Event wesentlich länger als der mittlere Triggerabstand für Änderungen zugänglich ist. Der Eventbuffer befindet sich im Arbeitsspeicher des GOOSY Event-Builders, der damit die Möglichkeit hat, mehrere Trigger zurückliegende Events zu ändern. Der Datenstrom vom MWDC Readout zum GOOSY Event-Builder enthält dann synchron zum Trigger Rohdaten und asynchron dazu die Ergebnisse der Kalibration, die in das Rohdaten-Event eingefügt werden.

Die Struktur für das MWDC Readout ist eine Pipeline von vier Stufen, von denen zwei parallelisiert sind. Den Aufbau der Pipeline mit den Kopplungsmechanismen zwischen den Stufen zeigt die Abbildung 4.1.

Die Aufgaben für die Prozessoren innerhalb der vier Verarbeitungsschritte sind:

X1, Y, X2 chamber:

do forever in all chambers
wait for event
wait till Event-Builder is ready to receive
read Hitpattern, number of first wire fired and TDC-values
assemble subevent
transport subevent to Event-Builder
end do

Event-Builder:

do forever
wait for and dequeue free event-buffer
wait for subevent of one chamber
assemble all other subevents into event
check event structure
acknowledge 'event received' to all chambers
enqueue dummy-event containing event-ID and length to Dumper
enqueue event to Analysis
end do

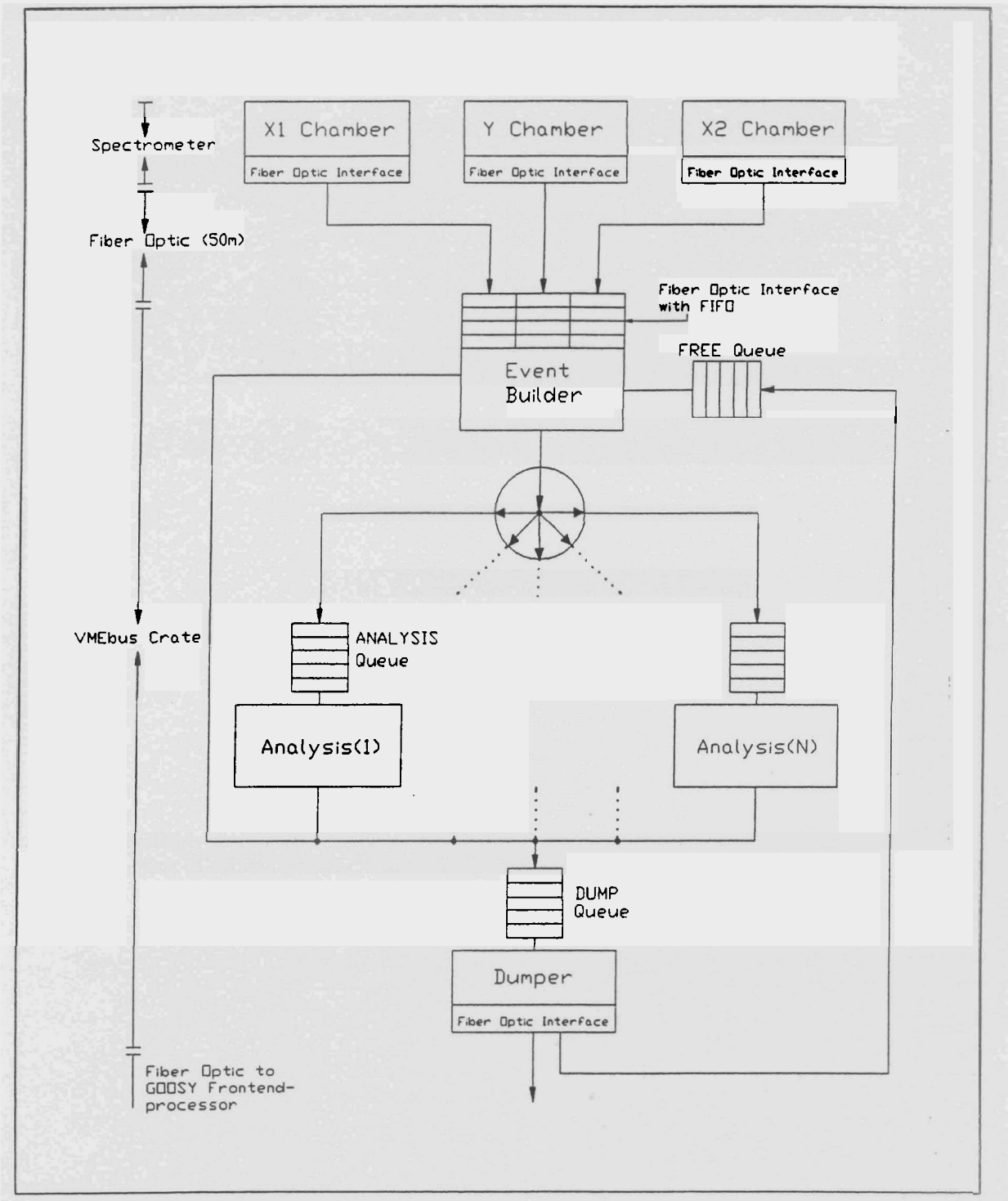

Abb. 4.1:
Struktur der Pipeline zur Ausführung der Kalibrationsalgorithmen. Die vier Auswertestufen sind von oben nach unten angeordnet. Ihre Standorte sind am linken Rand vermerkt. Der Fluß der Eventdaten ist mit Pfeilen markiert (siehe Kapitel 6.5).

Analysis (1,2,...):

do forever in all Analysis modules
wait for and dequeue event-buffer
apply calibration function
enqueue event to Dumper
end do

Dumper:

do forever
wait for and dequeue event-buffer
wait till GOOSY-Eventbuilder ready
transport event to GOOSY-Eventbuilder
enqueue event into free queue
end do

Die Hardwaremodule, die zur Umsetzung dieses Konzeptes einer Pipeline für das MWDC Readout notwendig sind, wurden im Rahmen dieser Arbeit entwickelt. Die Entwicklung wurde zwar vor der Festlegung dieser Rechnertopologie abgeschlossen, die Verwendung eines Bussystems zur Rechnerkopplung erübrigte aber Änderungen an der Hardware.

Die Entwicklung der insgesamt vier Module wurde einerseits von den MWDC spezifischen Anforderungen und andererseits von der Vorgabe bestimmt, das μP - Modul auch für andere Aufgaben anwenden zu können.

Das wurde erreicht durch die Umsetzung aller MWDC-spezifischen Anforderungen in der erste Stufe der Pipeline. Dadurch konnte das Mikroprozessor Modul zur folgenden Analyse als universell einsetzbare VME Central Processing Unit (CPU) entwickelt werden.

Von den vier Modulen setzen zwei einen Mikroprozessor ein. Ihre Zuordnung zu den Stufen der Pipeline ist:

- **X1,Y,X2:**
  Diese μP - Module sollen die Abgrenzung der Driftkammerelektronik von den nachfolgenden Stufen gewährleisten und den Übergang von den Digitalisieren auf eine serielle Datenübertragung über ein Lichtleiterpaar implementieren. Neben den Parallel/Seriell Wandlern enthält es einen μP, der in seiner Hauptschleife das Auslesen, Zusammenstellen und Übertragen der Kammerdaten übernimmt.

- **Event-Builder, Analysis, Dumper:**
  Das sind universelle VME CPU Platinen, die zusammen mit einem Ethernet Interface auch in andere Anwendungen eingesetzt werden. Das μP - Modul enthält insbesondere eine prozessorprivate Schnittstelle zu allen Interface Modulen, die nicht über den VME Bus angesprochen werden müssen, um den Systembus zu entlasten. Dazu gehört das Pendant des Lichtleiter Interfaces der Driftkammer-μPs und der Ethernet Controller. Der Übergang von Dumper zu GOOSY - Event-Builder benutzt ebenfalls die entwickelte serielle Verbindung über Lichtleiter, der zugehörige Übergang zwischen Lichtleiter und CAMAC ist in [20] beschrieben.

In den folgenden Kapiteln soll die entwickelte Hard- und Software zusammen mit ihrer Entwicklungsumgebung vorgestellt werden.

5 Hardware - Komponenten

Die Neuentwicklung der gesamten Hardware für das MWDC - Multiprozessor System erlaubte eine optimale Anpassung an die Problemstellung.

Es wurde versucht durch den Einsatz hochintegrierter (Very large Scale Integrated VLSI) Bauteile und programmierbarer Logikbausteine (Programmable Array Logic PAL) den Bauteileaufwand und damit den Aufbau- und Testaufwand zu senken. Grundsätzlich wurden möglichst viele Aufgaben in die Software ausgelagert, sofern sie nicht dadurch die Leistungsfähigkeit im Meßbetrieb beinträchtigen.

Die Schaltpläne wurden in drei Fällen Firmen übergeben, die die Entflechtung und Herstellung der Platinen übernahmen. Die VMEbus CPU Platine wurde im Rahmen dieser Arbeit auf einem Entflechtungssystem der Firma CALAY im Institut für Nachrichtentechnik der TH Darmstadt entwickelt.

Der Test der Platinen wurde durch den Einsatz eines 80 Kanal Logikanalysators der Firma Hewlett Packard HP 1650A erleichtert. Er ermöglicht neben der Timing Analyse auch das Disassemblieren der Prozessorzyklen.

Im folgenden werden die Platine durch ihre Funktionseinheiten aus der Sicht des Benutzers mit einigen Implementationshinweisen beschrieben.

5.1 VMEbus CPU

Die CPU besteht aus dem Prozessor MC68020 und dem Floatingpoint Koprozessor MC68881 oder MC68882 [10, 11]. Die eingesetzten Taktfrequenzen sind 12.5 Mhz für Steuerungsaufgaben und 25 Mhz für die Rechner des MWDC-Systems.

Der Datenbus zum lokalen Arbeitspeicher und zur VME Busschnittstelle ist 32 Bit breit. Die prozessorspezifische Verbindung zu den Interface Modulen ist 16 Bit breit. Sämtliche anderen Bausteine benutzen 8 Bit Datenbreite. Dieser lokale Daten- und Adressbus mußte nicht mit Treibern vom Prozessor abgekoppelt werden. Die prozessorinternen Treiber reichten für die Belastung durch die angeschlossenen Bausteine aus.

Der Arbeitsspeicher besteht aus EPROM (Electrically Programmable Read Only Memory), SRAM (Static Random Access Memory) und DRAM (Dynamic Random Access Memory). EPROM und SRAM benutzen ein 32-poliges Standardanschlußbild (JEDEC) für bausteine mit byteweiser Organisation. Die momentan eingesetzten Speichergrößen sind 256 kBit $\sim$ 4 * 64 kByte für das EPROM und 128 oder 512 Kbit $\sim$ 4 * 32/128 kByte für das SRAM. Die Wahl des genannten Anschlußbildes erlaubt den Einsatz von Speicherbausteinen verschiedener Generationen (128 Kbit, 256 kBit, 512 Kbit und 2048 Kbit) durch die Anpassung mit einigen Schaltern (Jumper) auf der Platine.

Ein optionaler Speicher von 1 oder 4 Mbyte benutzt byteweise organisierte DRAMs. Auch hier wurde das Anschlußbild so gewählt, daß Speicherplatzausbau möglich ist. Dieser Speicher wird momentan nur von intelligenten Grafikterminals [18] benutzt, die diese CPU enthalten.

Die Adresse, mit der der lokale Speicher über den VME Bus angesprochen wird, ist vom Benutzer durch die Wahl der Adressbits A24-A31 mittels Jumper festzulegen. Die Schalter A25 - A27 sind vom Prozessor lesbar. Sie legen eine Prozessor - ID (1 - 7) fest, die starr an die Platinenadresse gekoppelt ist.

Das VME Interface ist vom Typ A32 - D16/D32. Das bedeutet, daß jeder VME Zyklus alle 32 Adressbits und wahlweise eine Portgröße von 16 oder 32 benutzt. Da die VME Spezifikation keine Quittungsignale kennt, die es dem angesprochenen Modul erlauben, seine Portgröße zurückzumelden, wird diese Information aus der Portadresse gewonnen. Adressbit A24 legt fest, ob ein Langwort (A24 = 0) oder Wortzyklus vorliegt.

Das VME Interface ist Master/Slave fähig. Zugriffe, sowohl von der CPU zu anderen Modulen, als auch von anderen CPUs in den lokalen Speicher sind möglich. Aus schaltungstechnischen Gründen ist nur das SRAM über den VME Bus zugänglich. Interrupts von VME I/O Modulen werden auf Level 2 und 3 unterstützt und sind über Jumper zu selektieren.

Es wurde besonders auf die korrekte Implementation der ununterbrechbaren Zyklen geachtet. Sie stellt sicher, daß Datenstrukturen in einem Zugriff (der bis zu vier Zyklen umfassen kann) konkurrierend mit anderen Prozessoren geändert werden können.

Ein Interprozessor Interrupt (Level 4) wird über eine spezielle Addressierung des Zielmoduls ausgelöst, er wird automatisch gelöscht, wenn der Zielprozessor den Interrupt lokal quittiert.

Ein Dual Asynchronous Receiver Transmitter (DUART) MC 68681 [21] Multi Function Peripheral MFP MC 68901 [22] zusammen mit einem V24 Interface erlauben den Anschluß dreier Terminals. Sie benutzen Interrupt Level 1 und 5.

Die prozessorprivate Schnittstelle zu I/O Modulen hat eine Datenbreite von 16 Bit und einem Adressraum von 18 Bit. Alle Signale der MC68020 Busschnittstelle bis auf die Direct Memory Access (DMA) Steuerung stehen zur Verfügung. Diese Schnittstelle erleichtert die Entwicklung der I/O Module, da sie auf Grund der Ähnlichkeit mit dem Prozessorbus einfacher und mit weniger Aufwand als der VME Bus zu implementieren ist. Die Signale sind über Treiberbausteine vom Prozessorbus entkoppelt und werden über ein 64 - poliges Flachbandkabel zu den I/O Modulen geführt. Dabei wird der von der VME - Spezifikation unbenutzte Teil der Rückwandverdrahtung benutzt (P2-a/c).

Acht I/O Leitungen des MFP werden als Interrupt und Statussignale führ den prozessorprivaten Bus verwendet. Die Interrupts (Level 5) werden vom MFP vektorisiert, so daß auch Generation von Interrupts aus den I/O Modulen vereinfacht wird.

Die Platine selbst ist in 4-Lagen Multilayertechnik gefertigt. An die Frontplatte geführt sind ein Restart und ein Break Taster, die Terminal Schnittstelle und eine Anzeigeeinheit, die die Ausgabe einer Hexziffer erlaubt.

5.2 Driftkammer CPU mit Lichtleiter Interface

Ein Ziel der Entwicklung war die möglichst frühe galvanische Trennung der Driftkammerelektronik von den nachfolgenden Auswertestufen. Da die zu überbrückende Entfernung etwa 50 m beträgt, schied das Verlängern der kammerspezifischen Schnitt-

stelle (etwa 30 Signale) über Optokoppler aus. Die gewählte Methode ist ein serielles Hochgeschwindigkeits-Interface (6 Mbyte/sec) mit zwei Lichtleitern als Sende und Empfangsmedium.

Kernstück dieser Platine ist der Transparent Asynchronous Xmitter - Receiver Interface (TAXI) Chipsatz aus den Bausteinen AM 7968 und 7969 der Firma Advanced Micro Devices [23]. Sie enthalten die Parallel/Seriell Wandler, die Takterzeugung und Rückgewinnung und eine Übertragungsfehlererkennung. Diese Bausteine sind je nach Version bis zu 175 Mbit/sec Datenrate einsetzbar.

Die Sender und Empfangerbausteine können direkt über Koaxial Leitungen mit 50 Ohm Impedanz verbunden werden. Sie benutzten für die übertragung ECL Pegel. Alle anderen Signale haben TTL Pegel.

Das Parallel-Interface besteht auf beiden Seiten aus zwei Registern mit je einem 'Daten gültig' bzw. 'sende Daten' Signal. Das eigentliche Datenregister ist 8 - 10 Bit breit, das zweite bietet 4 - 2 Bit und soll z.B. für die Kontrolle des Datenflusses verwendet werden.

Die Driftzeiten sind der Hauptteil der auszulesenden und zu übertragenden Daten. Die TDCs haben eine Auflösung von 1 nsec. Die maximale Driftzeit ist ca. 220 nsec. Daher ist der zu erwartende Wertebereich für die gemessenen Driftzeiten auf 0 - 511 festgelegt worden. Der TAXI Chipsatz wird in diesem Modul dagegen mit 10 Bit Datenbreite betrieben. Es wird nur das Datenregister benutzt, dessen 10. Bit wird für Kontrollfunktionen verwendet. Die Datenrate beträgt 60 Mbit/sec.

Statt mit Hilfe dieser Parallel/Seriell Wandler den Driftkammerbus mit einer Steuerungslogik zu verlängern, die das Protokoll eines Prozessorzyklus simuliert, wurde zwischen dem Driftkammerbus und dem Lichtleiter-Interface ein Mikroprozessor eingefügt. Da er in die MWDC Pipeline integriert werden kann, erhöht das den Durchsatz, und der Driftkammerbus ist einfach in den Adressraum des Mikroprozessors aufzunehmen.

Der Prozessor MC 68020 wird mit einem nur 8 Bit breiten Arbeitsspeicher betrieben, um den Bauteileaufwand zu reduzieren. Der Arbeitsspeicher besteht einem EPROM (64 kByte) und einem SRAM (32 KByte). Nur die Verbindung zum Driftkammerbus ist 16 Bit breit.

Der hohe Zeitaufwand für alle Speicheroperationen durch das damit notwendige Multiplexen spielt während der Messung keine Rolle. Die Hauptschleife zum Auslesen und Übertragen der Driftkammerdaten paßt vollständig in das 256 Byte große Programm Cache. Der Cache Speicher ist den μP integriert und erlaubt schnellen Zugriff auf die zuletzt ausgeführten Befehle. Das bedeutet, daß während der Messung keine Speicherzugriffe außer zu den I/O Registern notwendig sind.

Die Definition des Driftkammer Bussystems und seine Adressbelegung durch die Konverter war vor der Entwicklung dieser Platine abgeschlossen. In diesem Modul wird der Adressraum des Driftkammerbus mehrfach in den des Mikroprozessors eingeblendet. Der gewählte Bereich legt die Transfergeschwindigkeit fest. Die Bussignale sind über Treiber mit der speziellen Rückwandverdrahtung der Driftkammer Elektronik verbunden.

Alle Timing und Steuersignale, (Hit, Strobe, Busy, Init, Clear) sind mit einem Stecker für Flachbandkabel an die Frontplatte geführt. Alle Steuersignale der drei Kammern

werden so mit einem Kabel verbunden und an einen speziellen Leveladapter angeschlossen, der einge logische Verknüpfungen vornimmt und den Übergang zu einer Standard NIM Triggerlogik erlaubt.

In enger Verbindung zu den genannten Steuersignalen steht ein Statusregister des Prozessors, das folgende Information enthält:

'Busy': Das Bit wird gesetzt von der Steuerlogik, wenn mit den Signalen der Kammern und des Szintillators ein gültiges Ereignis erkannt wurde. Der Prozessor wartet in seiner Ausleseschleife auf das gültige Busy-Bit und setzt es zurück nach der Übertragung des Events.

'NN': Ein freies Bit, das von der Steuerungslogik kontrolliert wird

'receiver ready': Dieses Bit wird automatisch gesetzt, wenn ein Datenwort über den Lichtleiter empfangen wurde und gelöscht, wenn es gelesen wurde.

Ein weiteres Statusregister, das direkt über das Lichtleiter Interface ohne Beteiligung des lokalen Prozessors gesetzt werden kann, kontrolliert den Prozessor-Restart und erlaubt es, Interrupts auszulösen.

Die Platine ist doppelseitig und enthält einen besonders abzuschirmenden Bereich, in dem sich die optoelektronischen Bauelemente und ein Operationsverstärker des Empfangsteiles befinden.

5.3 Lichtleiter Interface zu den VMEbus CPUs

Die Verbindung zu den Driftkammern ist während der Messung stark belastet. Da ein Event-Builder eingesetzt wird, der alle Driftkammern von den folgenden Pipelinestufen abkoppelt, ist es nicht notwendig die Driftkammern über ein VME Interface anzusprechen. Daher kann der prozessorprivate Bus der entwickelten VMEbus CPUs benutzt werden. Dadurch wird die Belastung des VME Bus als Medium für die Interprozessor Kommunikation innnerhalb der folgenden Pipelinestufen verringert.

Das Module besteht im wesentlichen aus den gleichen Parallel/Seriell Wandlern mit zugehöriger Optoelektronik, wie sie auch in den Driftkammer - Auslesemodulen verwendet werden. Die Empfängerseite wird ergänzt durch ein 512 Worte tiefen FIFO (First In First Out) Zwischenspeicher. Das FIFO erlaubt eine Organisation des Datenflusses zwischen den Ausleseprozessoren der Driftkammern und dem Event-Builder, dessen Overhead vernachlässigt werden kann.

Das Statussignal 'FIFO - empty' dient entweder als Interruptquelle oder als Statusbit innerhalb des MFP Bausteines der VMEbus CPU, der die Statussignale des privaten Busses verwaltet.

Zusätzlich wird die asynchrone Betriebsart der MC 68020 für Speicherzugriffe ausgenutzt. In dieser Betriebsart dauern Speicherzyklen so lange bis ein Quittungssignal erzeugt wurde. Damit wird das Lesen des FIFOs so lange verzögert bis mindestens ein Wort im FIFO enthalten ist.

Die Platine ist ebenfalls doppelseitig, einige Teile sind identisch zu der Driftkammer - Ausleseprozessor Platine.

5.4 Ethernet Interface

Ein Rechnernetz ist die Verbindung mehrerer Computer über ein Medium, das ihnen Informationsaustausch ermöglicht. Die wesentlichen Merkmale eines Rechnernetztes sind: die Datenrate, die Struktur der Datenpakete, die möglichen Topologien der Verbindung, das Medium, welche Entfernungen überwunden werden können und von wie viele verschiedene Rechner- bzw. Betriebssysteme eingebunden werden können.

Ethernet ist eine Definition [24] eines Rechnernetzes, das in allen diesen Punkten den Anforderungen eines Rechnerverbundes im Beschleunigerlabor erfüllt. Ethernet wird zur Verbindung der Institutsrechner zu einem VAX-Cluster und als unterste Transportschale in verschiedensten Protokollen, die den hochschulweiten Rechnerverbund garantieren.

Die Datenpakete sind maximal 1500 Bytes lang. Sie enthalten die Sender und die Zieladdresse, einen Pakettyp, die eigentliche Benutzerdaten und eine Prüfsumme zur Aufdeckung von Übertragungsfehlern. Die Adressen sollten eindeutig einem Interface zuzuordnen sein. Es sind aber Adresstypen (Broadcast, Multicast) vorgesehen, die alle oder eine Gruppe von Interfaces mit dem gleichen datenpaket adressiert. Die Übertragungsrate ist 10 Mbit/sec auf einem Koaxial Kabel, dessen maximale Länge ist etwa 1500m. Auf der ganzen Kabellänge können mit speziellen Kopplern Ethernet Interfaces angeschlossen werden. Dadurch ist zwangsläufig jeder Rechner mit allen anderen verbunden.

Die damit notwendige Zugriffskontrolle muß vom Interface übernommen werden und ist Teil der Ethernet-Definition. Diese Methode heißt 'Carrier Sense Multiple Access with Collision Detection' CSMA/CD.

Mit dieser Methode wartet das Interface nach einem Sendeauftrag vom Rechner bis kein Teilnehmer mehr sendet und beginnt dann mit der eigenen Übertragung. Beginnen mehrere Sender gleichzeitig, kommt es zu Störungen (Collisions), die von allen Sender erkannt werden müssen, um unmittelbar danach das Senden einzustellen. Danach wartet jeder unterbrochene Teilnehmer eine Zeitspanne, die mit Hilfe eines vorgeschriebenen Algorithmus und eines Pseudozufallszahlen Generators ermittelt wird. Dadurch wird die erneute Gleichzeitigkeit des Sendens verhindert.

All diese Funktionen werden üblicherweise von VLSI Bausteinen übernommen, die das gesamte Ethernet-Protokoll übernehmen und direkten Zugang zum Arbeitsspeicher des Computers haben. Sie können so empfangene Packete ohne Beteiligung des Rechners ablegen und selbstandig die Daten nach einem Sendeauftrag auslesen.

Das entwickelte Interface verwendet den Local Area Network (LAN) Coprozessor i82586 der Firma Intel [26]. Er benutzt ein Speichermodell von bis zu 2^{24} Bytes, das in 64 kByte Segmente unterteilt ist. Die Datenbusbreite kann 8 oder 16 Bit sein und wird in der Initialisierungsphase festgelegt.

Der Baustein besitzt keine eigenen Register zur Steuerung. Der Anschluß an den Hauptprozessor benutzt nur jeweils eine Interruptleitung von und zu dem Koprozessor. Der gesamte Befehls und Datenaustausch benutzt mehrere verkettete Listen von Speicherblöcken, auf die der LAN Controller über DMA zugreift.

Das entwickelte Modul hat daher als zweite wesentliche Komponente einen eigenen Ar-

beitspeicher der Größe von 32 kWords. Sowohl der Prozessor als auch der Koprozessor können konkurrierend auf diesen Speicher zugreifen.

Bei der entwickelten Speicherzuteilungslogik wird die Möglichkeit beider Mikroprozessoren CPU und LAN Controller genutzt, einen Speicherzyklus so lange zu verzögern, bis ein Quittungsignal gegeben wird.

Der Schnittstelle zu den VMEbus CPUs ist wieder deren privater Peripheriebus, der mit seinem 18 Bit Adressraum genügend Platz für den Speicher bietet, der mit dem LAN Controller geteilt werden soll. Das Interruptsignal zum Hauptprozessor benutzt wahlweise eine acht Leitungen des Peripheriebusses. Das Signal vom Hauptprozessor zum LAN Controller, einen Befehl abzuarbeiten, wird ausgelöst, wenn eine bestimmter Adressbereich des Ethernet Interfaces angesprochen wird.

Die Taktrückgewinnung und eine Pegelanpassung an das Thickwire-Kopplerinterface ergänzen die Schaltung.

Die Platine ist doppelseitig und enthält Schaltungsteile, die den Betrieb des Interfaces mit einem eigenen MC 68000 Prozessor zur Verwaltung des LAN Controllers erlauben. Um die Entwicklung des Ethernet-Gerätetreibers für den Multitasking Kern zu vereinheitlichen, wurde diese Abgrenzung der Controller spezifischen Funktionen von dem Hauptprozessor aufgegeben.

6 Software - Komponenten

Die Entwicklungsumgebung auf den Institutsrechnern, einem VAX-Cluster, wurde aus folgenden Komponenten zusammengestellt:

- C - Compiler, Assembler, Linker und Librarian der Firma Microtec Research.
- Multiprozessorfähiger Multitasking Kern pSOS^{+m} und Debugger pROBE^{+} der Firma Software Components Group.
- Code Management System CMS und Module Management System MMS der Firma Digital Equipment als Teil des vorhandenen Paketes für die VAX - Softwareentwicklung.

Die Wahl der Programmiersprache C ist auf das Angebot im Cross-Software Markt und einem de-facto Standardeinsatz der Sprace C in hardwarenahen Problemen zurückzuführen. FORTRAN wäre zumindest als Ergänzung wertvoll, da die Entwicklung der numerischen Methoden zur Umkehrung der Spektrometerabbildung [27] in FORTRAN erfolgt.

Der Multitasking Kern pSOS^{+m} [13] bietet die Möglichkeit, alle Methoden der Intertask-Kommunikation über Prozessorgrenzen hinweg zu benutzen. Der Hersteller legt das Medium für die Prozessorverbindung nicht fest. Es können beliebige Topologien verwirklicht werden. Diese Flexibilität erfordert vom Entwickler eines Rechnernetzes auf Grundlage von pSOS^{+m} die Erstellung eines eigenen Kommunikationsprotokolles, das die Verbindungen zwischen den Rechner herstellt.

CMS erlaubt die Archivierung, Versionskontrolle und Sicherung von getesten Programmen, die beteiligten Module können nach fehlerhaften Änderungen wieder auf den Stand der gesicherten Version gebracht werden. Gleichzeitige Änderungen eines Moduls durch mehrere Entwickler können kanalisiert werden.

MMS verwendet eine Liste aller Äbhängigkeiten zwischen den Quelldateien und ein Regelwerk, das sämtliche Schritte des Entwicklungszyklusses beschreibt, zur automatischen Erstellung des Gesamtprogrammes. Bei einer Neuerstellung des Programmes werden nur Module übersetzt, die geändert wurden.

Alle Programme innerhalb der MWDC Pipeline wurden in der Sprache C kodiert. Bibliotheksfunktionen und Gerätetreiber, die den Anschluß an die Hardware bilden, sind in Assembler als C - Unterprogramme kodiert.

Einige Anpassungen des Multitasking Kerns pSOS^{+m} und des Debuggers pROBE^{+} insbesondere die Verbindung der beteiligten Rechner erforderten Assemblerprogramme, die der Aufrufkonvention des Herstellers folgen müssen. Auch der komplette Code der Ausleseprozessoren der Driftkammern wurde in Assembler erstellt.

6.1 pSOS^{+m} Kernel Interface

Damit kennzeichnet der Hersteller die Schnittstelle zwischen dem pSOS^{+m} Kern und dem dem Kommunikationsmodul, das vom Entwickler erstellt werden muß, um die

Verbindung zwischen den Prozessoren des Systems zu gewährleisten. Durch diese Abgrenzung ist das Medium zur Datenübertragung und die Topologie der Verbindungen variabel.

Der Entwickler muß dem Kern einen Satz von Funktionen zur Verfügung stellen, die den Anforderungen des Kernel Interface (KI) genügen. Die sichere Übertragung der Pakete, die Erhaltung ihrer Reihenfolge und der vollständige Aufbau der Verbindungen während des Starts des Kerns muß garantiert werden. Die gefordeten Funktionen sind in der folgenden Tabelle zusammengefaßt.

Tabelle 6.1: Funktionsumfang des pSOS^{+m} Kernel Interface

Name	Funktion
KI_INIT	**koordiniert** den gemeinsamen Start der Prozessoren des Systems. Sämtliche Verbindungen müssen aufgebaut werden, danach kann kein Prozessor in das System aufgenommen werden. Alle folgenden Funktionen müssen danach zur Verfügung stehen.
KI_GETPKB	dient pSOS^{+m} dazu, einen Speicherbereich zu reservieren, der dann mit dem zu übertragenden Paket ausgefüllt wird.
KI_RETPKB	erlaubt es pSOS^{+m}, ein Datenpaket zurückzugeben.
KI_SEND	sendet ein Paket an den Zielrechner.
KI_BROADCAST	sendet ein Paket an alle Rechner des Systems.
KI_RECEIVE	liest ein empfangenes Paket in einen mit KI_GETPKB allokierten Speicherbereich ein. Dem Kern wurde vorher mit dem Aufruf der Funktion KI_ANNOUNCE der Empfang eines Paketes mitgeteilt.

Das MWDC-System verwendet geteilte Speicherbereiche und serielle Datenübertragung zur verbindung der Prozessoren. Die Lichtleiter-Interfaces sind jeweils nur einem Prozessor zugänglich. Die Topologie des erstellten Kernel Interface ist damit:

- Jeder VMEbus-Rechner kann mit bis zu vier Rechnern über Lichtleiter verbunden sein. Das können Standalone VME oder Driftkammer CPUs sein.
- Maximal sieben VMEbus CPUs sind innerhalb des VME Crates über geteilte Speicherbereiche gekoppelt.
- Pakete, die eine Lichtleiterverbindung benutzen müssen, werden von der VMEbus CPU weitergeleitet, das das zugehörige Lichtleiter-Interface bedient.

Die Funktionen werden im Kontext des pSOS^{+m} Kerns oder einer Interrupt Service Routine ausgeführt. Dabei wird der Interprozessor-Interrupt der VME Boards und der 'FIFO-not empty' Interrupt des Lichtleiter-Interfaces benutzt. Der Funktionsumfang wurde um Sende und Empfangsroutinen für Pakete erweitert, die Debugger Informationen enthalten (siehe Kapitel 6.2).

Diese KI-Implementation verwendet keine Funktionen des Kerns. Sie ist damit auch vor dem Start von pSOS^{+m} während der Initialisierungsphase des Systems benutzbar.

6.2 System Level Debugger pROBE^{+}

Dieser Debugger ist eine Erweiterung von pSOS^{+m} [14]. Er hat Zugang zu allen Kernstrukturen, so daß Kontrolle der Systemobjekte, Breakpoints auf Kernaufrufe oder Änderungen des Systemzustandes zusätzlich zu allen normalen Debuggerfunktionen von der Konsole aus möglich sind. Damit sind alle Zustandsänderungen des Systems durch die Nutzung des Multitasking Kerns beobachtbar bzw. zu stimulieren.

Leider ist dieser Debugger, obwohl als Ergänzung des multiprozessorfähigen pSOS^{+m} angeboten, völlig auf den lokalen Rechner beschränkt. Laden und Testen der Programme für ein Multiprozessorsystem ist nur mit vielen Terminals möglich. Fur jeden Rechner des Gesamtsystem existierten dann völlig unabhängige Testumgebungen, ohne daß die Verkopplung durch das pSOS^{+m} berücksichtigt würde.

Deshalb wurde das oben beschriebene Kernel Interface erweitert, um die Kommunikation zwischen den beteiligten Debuggern zu ermöglichen. Dabei wurde ein neuer Pakettyp und zusätzliche Sende und Empfangsroutinen eingeführt.

Wie bei pSOS^{+m} sind vom Anwender einige Funktionen zu entwickeln, die im Kontext des pROBE^{+} oder pSOS^{+m} aufgerufen werden, um die Anbindung an das Rechnersystems herzustellen (Callback-Funktionen). Callbacks sind z.B Routinen für Console-I/O, Symbolübersetzung, Gerätetreiber und unter anderen auch eine Routine, die vom Debugger aufgerufen wird, wenn eine Eingabezeile kein bekanntes Kommando enthält. Der Benutzer hat dann die Möglichkeit, die Kommandozeile zu interpretieren.

In dem Kontext der Callbacks 'Console-I/O' und 'User defined Command' und der Interrupt Service Routinen der Hardware zur Interprozessor-Kommunikation wurde folgender Funktionsumfang mit Hilfe des erweiterten Kernel Interfaces implementiert:

- **TO** n 'line' (n = Prozessor-ID)
  Sendet die Kommandozeile zu dem Debugger des Zielrechners und gibt dessen Antwort aus.

- **BREAK** n
  Sendet ein Break-Signal and den Zielrechner und gibt dessen Antwort aus. Die Interrupt Service Routine, die von der Hardware zur Interprozessor-Kommunikation ausgelöst wird, uberträgt dabei die Programmkontrolle an die Break Routine des Debuggers.

- **RL** n
  Lädt ein Programm vom Entwicklungsrechner über eine zweite Terminalschnittstelle in den Zielrechner. Die Zuordnung von Prozessor-ID zu Filenamen ist durch das DCL-Symbol 'LOAD_n' gegeben. Es enthält den DCL-Befehl 'TYPE Filename'. Dieses Symbol wird als Befehlszeile am Anfang der übertragung an den Entwicklungsrechner gesendet, der daraufhin das Programm an den Debugger sendet.

- **RESTART**
  Startet das gesamte System neu. Dabei wird ein auf allen Rechnern ein Kaltstart simuliert.

- **FIRST** 'line'
  Nach dem nächsten Restart wird diese Zeile vom Debugger als erste ausgeführt. Automatischer Start des Systems ist damit möglich, ohne die Eingriffsmöglichkeiten 'durch den Debugger zu verlieren. Die Gültigkeit der abgespeicherten Zeile wird mit einer Prüfsumme getestet.

Diese Befehle stellen insbesondere den Neustart und die Fehlersuche in den Driftkammerrechnern sicher, die sich auf dem Spektrometer befinden und keine eigene Terminalschnittstelle besitzen.

6.3 Ethernet Gerätetreiber

Die Konzeption der S-DALINAC Steuerung verlangt von jeder Steuer- und Meßeinheit die Integration in das Institutsrechnernetz. Das bedeutet, daß die Peripheriegeräte an Rechner mit Zugang zu dem Ethernetstrang des Institutes angeschlossen sein müssen. Die Programmpakete, die die Transportschale der Steuerungssoftware implementieren, sind DECnet und das Linac Control Protocol (LINCP) [28], das den Steuerrechner des Strahlführungsystems (LSI 11/73) in eine graphische Benutzeroberfläche auf VAX-Workstations [15] einbindet.

Um die im Rahmen dieser Arbeit entwickelte VMEbus CPU in diesen Steuerungskonzept einzubinden, mußte neben dem in Kapitel 5.4 vorgestellten Ethernet Interface auch der zugehörige Gerätetreiber für den Multitasking Kern entwickelt werden.

Der Kern definiert als Schnittstelle zu einem Peripheriegerät folgende Aufrufe:

Tabelle 6.2: I/O Kernaufraufe in pSOS^{+m}

Funktion	Aufgabe
DE_INIT	Gerät und Treiber initialisieren
DE_OPEN	Gerät für folgende I/O Operationen öffnen bzw.
DE_CLOSE	schließen
DE_READ	Datensatz lesen
DE_WRITE	Datensatz schreiben
DE_CONTROL	alle weiteren Gerätefunktionen

Die Parameter aller dieser Funktionen sind eine Gerätenummer, ein gerätespezifischer Parameterblock, und ein Returncode. Der Kern leitet diese Aufrufe unmittelbar über eine Tabelle für jedes Gerät in Prozeduren um, die vom Entwickler zu stellen sind. Weitere Hilfen als die Festlegung des Funktionsumfanges und der Parameterübergabe sind nicht vorgesehen.

Allerdings werden diese Prozeduren (Init, Open, ...) im Kontext der aufrufenden Task ausgeführt, daher können sämtlich Kernaufrufe benutzt werden. Auch in den Interrupt Service Routinen können Kernfunktionen aufgerufen werden. Sie dürfen aber nicht versuchen zu blockieren (z.B Wait for Event). Das pSOS^{+m} dient also nur dazu, eine einheitliche Aufrufkonvention für die Prozeduren zur Ein und Ausgabe anzubieten.

Der entwickelte Ethernet Treiber folgt dieser Konvention. Er implementiert das 'extendend IEEE 802.3' [25] Ethernet Protokoll und erlaubt mehreren lokalen Tasks die Benutzung des Ethernets.

Das 'extended IEEE 802.3' Ethernet Protokoll überträgt zusätzlich zur Benutzerinformation Quelladresse, Zieladresse (jeweils 6 Byte), einen festen Protokolltyp (2 Byte), einen variablen Protokolltyp (5 Byte), die Paketlänge (2 Byte) und eine Prüfsumme. Während die Adressen die beteiligten Rechner identifizieren, ist der Protokolltyp eine Identifikation der verschiedenen Programmpakete, die das Ethernet als Transportmedium benutzen. Der Gerätetreiber hat dadurch die Möglichkeit ohne Kenntnis dieser Programme nur mit Hilfe des Protokolltyps die empfangenen Datenpakete an die einzelnen Benutzer (Tasks) weiterzuleiten. Mit jedem angemeldeten Protokolltyp wird

durch die Open-Funktion wird daher ein neues Ethernet Device im Sinne von pSOS^{+m} erzeugt (minor device).

Nach dem Öffnen (DE_INIT) des Kanals werden automatisch alle Pakete an den eigenen Rechner mit dem angegeben Protokolltype in einer pSOS Queue zwischengespeichert (bis zu einer beim Open angegebenen Maximalzahl).

Die Task kann nach erfolgreichem Auslesen der Paketkennung aus dieser Queue das Datenpaket vom Speicher des Ethernet Controllers in einen eigenen Bereich einlesen (DE_READ).

Das Senden eines Paketes (DE_WRITE) blockiert die aufrufende Task nicht, die Daten werden sofort in den Speicher des Ethernet Controllers übertragen und der Sendebefehl in seine Kommandoliste eingefügt.

Multicast Adressen, die der Ethernet Controller erkennen soll, werden mit der Funktion DE_CTRL angegeben. Danach muß jede Task einzeln Multicasts oder den Broadcast unter Bezug auf diese Liste freigeben, sofern ihr Kanal (Protokoll) diese Adressierungsart benutzen soll.

6.4 Einbettung der MWDC - Pipeline in pSOS^{+m}

Die MWDC Pipeline verwendet für den Datenfluß während der Messung keine Funktionen des Kerns. Das hat zwei Gründe:

Die sonst sehr schnellen Ausführungszeiten des Kerns (20 - 70 μsec für einen 16 Mhz MC 68020 [29]) sind bei Benutzung der Systemaufrufe über Prozessorgrenzen hinweg stark verlängert. Außer dem inherenten Aufwand durch durch die eigene Implementation des Kernel Interface trägt dazu wesentlich das zweimalige Rescheduling bei Benutzung eines Systemaufrufs bei, auch wenn der Task bei lokaler Ausführung nicht blockieren würde. Das Rescheduling, also die Auswahl des nächsten wartenden Tasks, der Rechenzeit bekommen soll, wird dadurch verursacht, daß jeder Systemaufruf die Ergebnisse seiner Ausführung auf dem Zielrechner abwarten muß und daher blockiert wird.

Der zweite Grund ist, daß die Mailbox (Queue) innerhalb von pSOS^{+m} eine Maximalgröße von 16 Bytes hat. Diese Einschränkung ist bei Systemen, die mit geteilten Speicherbereichen gekoppelt sind, zu umgehen, in dem die Adressen der zu übertragenden Daten als Nachricht verwendet werden. Bei einer Rechnerkopplung über ein anderes Medium ist das unmöglich. Dieser klare Designfehler von pSOS^{+m} verhinderte den Einsatz von pSOS^{+m} für die Verbindung der Pipeline-Stufen.

Die Initialisierung und Überwachung der einzelne Module (X1, Y, X2, Event-Builder, Analysis, Dumper) dagegen nutzte viele Möglichkeiten von pSOS^{+m}.

Dazu sind diese Module als Tasks mit der jeweils niedrigsten Priorität gestartet. Diese Tasks warten auf ihre Eingabedaten, ohne im Sinne des Kerns zu blockieren und damit eine Kontextwechsel zu verursachen (polling mode). Trotzdem bekommen Tasks mit höherer Priorität sofort Rechenzeit, wenn sie bereit sind (preemptive scheduling).

Die verwendeteten Methoden des Kerns sind:

- **Asynchronous Signals**
  Die Kontrolle über den Zustand dieser Tasks und ihrer privaten Datenstrukturen

wird erleichtert durch die Möglichkeit, jedem Task eine Software-Interrupt Service Routine 'Asynchronous Signal Routine' (ASR) zuzuordnen, die die Programmkontrolle übernimmt, wenn ein 'Asynchronous Signal' an diese Task geschickt wurde und sie selbst aktiv ist. Diese Signale werden für Start, Restart Readout, Stop usw. verwendet.

- **Queues**
  Diese Form einer Mailbox mit einer festen Nachrichtengröße von 16 Bytes wird in der Initialisierungsphase der Tasks verwendet, um Konfigurationsparameter auszutauschen. Die Schwellwerte der Diskriminatoren z.B. werden so den Driftkammerprozessoren übermittelt.

- **Task - Register**
  Das sind 16 beliebige Langworte, die einer Task zugeordnet sind und von jeder anderen Task gesetzt und gelesen werden können. Sie dienen dazu, die Adressen von Variablen, die zur Laufzeit bestimmt werden müssen, bekannt zu machen. Das wird bei der Verbindung der Pipeline Stufen innerhalb des VME System genutzt, um die Queue-Kontrollstrukturen zugänglich zu machen. Sie werden im lokalen Speicher der zugeordneten Stufe eingerichtet (z.B Dumper-queue im Speicher des Dumper-Rechners) und dann über die Task-Register bekannt gemacht.

Alle Pipeline Stufen innerhalb des VME Crates führen nach dem Systemstart identischen Code aus. Die zuerst gestarte Task (ROOT) stellt mittels der angeschlossenen Hardware den Typ fest und erzeugt die zugehörige Task:

- Ist ein Lichtleiterinterface vorhanden, aber nicht vom Kernel Interface benutzt, da es währen des Starts des KI wegen fehlender Antwort eines Rechners ausgetragen wurde, so ist es das Dump - Modul.

- Sind Lichtleiterinterface(s) vorhanden, die zu Driftkammerrechnern führen ist es der Event-Builder.

- alle anderen VMEbus CPUs sind automatisch vom Typ Analyse.

Der Task der Driftkammerrecher wird direkt nach dem Systemstart erzeugt und wird über eine ASR und Queue kontrolliert.

6.5 Datenflußkontrolle im MWDC System

Das vorige Kapitel erläuterte, warum pSOS^{+m} nicht zur Verbindung der einzelne Stufen der Pipeline benutzt wird.

Die entwickelte Lösung verwendet eigene Queues zwischen allen Stufen, um deren zeitliche Entkopplung zu erreichen. Eine Stufe kann damit ihre Ergebnisse in der folgenden Queue abspeichern, ohne darauf zu warten, daß die nächste Stufe bereit ist zur Übernahme.

Die Queues zwischen den Stufen innerhalb des VME Systems (Event-Builder, Analysis, Dumper) sind einfach verkettete Listen ihrer Elemente (Eventdaten) deren Anfangs

und Endelement im Queue-Head referenziert werden. Die verwendeten Operationen auf diesem Datentyp sind 'Einfügen am Ende' und 'Auslesen am Anfang' der Queue.

Wie in der Abbildung 4.1 angedeutet, bilden die Queues 'Free', 'Analyse' und 'Dumper' einen geschlossenen Kreislauf. Ist die Eventrate so hoch, daß die Analyse-Stufe oder der GOOSY Frontendprozessor nicht schnell genug sind, werden alle Elemente innerhalb dieses Kreislaufes aus der 'Free' Queue entfernt, und der Event-Builder kann nicht weiterarbeiten.

Start und Stop des Datenflußes hängen also von den Verbindungen zu den Driftkammerrechnern und dem GOOSY Frontendprozessor ab. Die Verteilung auf die parallel arbeitenden Analysestufen dagegen wird von den Event-Datenstrukturen selbst übernommen. Sie werden während der Initialisierungsphase von den Analyse-Rechnern in deren lokalem Speicher erzeugt und erhalten Referenzen auf ihre eigene 'Analyse' und die 'Free' und 'Dumper' Queue-Heads. Die Elemente werden dann vom Event-Builder so sortiert, daß Events des gleichen Analyse-Rechners maximalen Abstand haben.

Während der Messung ist dann der Weg der Eventdaten festgelegt. Die Queue der jeweils folgenden Stufe ist in der Eventdatenstruktur eingetragn. Die anfängliche Sortierung garantiert die gleichmäsige Auslastung der Analysestufen.

Die zweite Methode der Datenflußkontrolle dient den Verbindungen Driftkammerrecher zu Event-Builder und Dumper zu GOOSY Frontendprozessor. Da in beiden Fällen die Queues in den FIFOs der Lichtleiter-Interfaces enthalten sind, wäre die Maximalzahl aller Events, die sich gleichzeitig im System befinden dürfen, durch die Speichertiefe der FIFOs begrenzt, wenn der oben beschriebene Kreislauf auf das gesamte System ausgedehnt würde. Denn da jede Stufe blockieren kann, muß die vorgeschaltete Queue alle Events aufnehmen können, bis die steuernde 'Free' Queue leer ist. Daher kontrollieren die beiden Sender, also Driftkammerrechner oder Dumper, den Datenfluß mit einer anderen Methode.

Beide Stufen, X1,Y,X2 und Dumper, erhalten am Start der Messung die Erlaubnis, eine Zahl N von Events zu senden, ohne daß eine Quittierung abgewartet werden muß. Ist dieser Kredit durch das Senden verbraucht, wird gestoppt. Der Empfänger schickt für jedes Paket eine Quittung zurück, die den Kredit wieder inkrementiert. Ist die Zeit vom Absenden bis empfangener Quittung für ein Event kleiner als die Zeit zum Senden aller N Events, muß nie gestoppt werden auf Grund des Quittungsbetriebes.

7 Status

Sämtliche Hardware-Komponenten VMEbus CPU, Driftkammer CPU, Lichtleiterinterface für die VMEbus CPUs und das Ethernet Interface sind aufgebaut und erfolgreich getestet. Sie werden eingesetzt in:

- der Steuerung und Überwachung der Hochrequenzregelung des S-DALINAC [16]
- der Steuerung der Elektronenkanone des S-DALINAC [19]
- einem Eingabegerät für die Beschleunigersteuerung, das Drehknöpfe als Einstellungshilfen für beliebige Geräte anbietet. Diese Einheit ist über Ethernet mit allen Stellrechnern und Workstations verbunden [17]
- einem intelligenten Graphikterminal [18], das auch als Ein- und Ausgabegerät für das Graphische Kern System GKS verwendet werden kann.

Schon unmittelbar nach dem Aufbau der Hardware konnten mit einem einfachen Programmsystem Messungen an jeweils einer der neuen Driftkammern durchgeführt werden [30, 18]. Dieses Programm erlaubte das Auslesen und die Steuerung einer Driftkammer und das Aufzeichen der Daten über eine serielle Schnittstelle.

Das gesamte MWDC System ist in der Pipeline Konfiguration aufgebaut und erfolgreich in GOOSY integriert. Messungen zur Untersuchung der Eigenschaften der Driftkammern konnten mit einer ^{90}Sr-Quelle (370MBq) an Stelle eines Streutargets und dem Q-Clam Spektrometer durchgeführt werden [2].

Der Durchsatz mit einer Analyse im Leerlauf liegt bei 7 kHz. Sie wurde ohne Datenflußkontrolle in dem Dumper und automatischer Triggererzeugung in den Driftkammer CPUs mit einem Ratemeter gemessen. Die Totzeit innerhalb der Driftkammerrechner beträgt dabei 12 μsec.

Die Entwicklung der Algorithmen zur Invertierung der Spektrometerabbildung [27] befindet sich im Abschluß, ihre Tests und Verbesserungen werden Offline auf dem Institutsrechner durchgeführt, um die notwendigen Strahlzeiten am S-DALINAC optimal auszunutzen. Daher konnten die Algorithmen noch nicht in das MWDC-System aufgenommen werden.

Literaturverzeichnis

[1] K. Alrutz-Ziemssen, D. Flasche, H.-D. Gräf, V.Huck, M. Knirsch, W. Lotz, A. Richter, T. Rietdorf, P. Schardt, E. Spamer, A. Staschek, W. Voigt, H. Weise and W. Ziegler, Proc. 4^{th} Workshop on RF Superconductivity, KEK **89-21**, Tsukuba, Japan (1989) 53.

K. Alrutz-Ziemssen, D. Flasche, H.-D. Gräf, V.Huck, M. Knirsch, W. Lotz, A. Richter, T. Rietdorf, P. Schardt, E. Spamer, A. Staschek, W. Voigt, H. Weise and W. Ziegler, Proc. Part. Acc. Conf., Vol **29**, (1990) 53

[2] K.D.Hummel, Dissertation, TH Darmstadt, in Vorbereitung

[3] G.Küchler, Dissertation (1986), TH Darmstadt

[4] H.D.Gräf, H.Miska, E.Spamer, O.Titze and Th.Walcher, Nucl. Inst. Meth. **153** (1978) 9

Th.Walcher, R.Frey, H.D.Gräf, H.Miska, E.Spamer, and H.Theissen, Nucl. Instr. Meth. **153** (1978) 17

D.Schüll, J.Foh, H.D.Gräf, H.Miska, R.Schneider, E.Spamer, H.Theissen, O.Titze, and Th.Walcher, Nucl. Instr. Meth. **153** (1978) 29

J.Foh, R.Frey, R.Schneider, D.Schüll, A.Schwierczinski, H.Theissen and O.Titze, Nucl. Instr. Meth. **153** (1978) 43

[5] G.Herbert, Dissertation, TH Darmstadt, in Vorbereitung

[6] F.Sauli, Principles of Operation of Multiwire Proportional and Drift Chambers, CERN 77-09 (1977)

[7] CAMAC Updated specifications (EUR 4100, 4600, 6100, 6500), Esone Comittee (1983)

[8] R.Amend, Diplomarbeit (1988), TH Darmstadt

[9] K.D.Hummel, private Mitteilung (1988)

[10] MC 68020 32-Bit Microprocessor User's Manual, Motorola

[11] MC 68881 Floating-Point Coprocessor User's Manual, Motorola

[12] "The VMEbus Specification", IEEE-1014 or IEC-821

[13] pSOS$^+$/68K User's Manual, Software Components Group

[14] pROBE$^+$/68K User's Manual, Software Components Group

[15] G.Kalisch, Diplomarbeit (1990), TH Darmstadt

[16] D.Flasche, Dissertation (1989), TH Darmstadt

[17] K.D.Hummel, private Mitteilung (1990)

[18] J.Horn, Diplomarbeit, TH Darmstadt, in Vorbereitung

[19] J. Töpper, private Mitteilung (1990)

[20] R.Kämpf, Diplomarbeit, TH Darmstadt, in Vorbereitung

[21] MC 68681, product specification, Motorola

[22] MC 68901, product specification, Motorola

[23] TAXIchip Integrated Circuits, data sheet, Advanced Micro Devices

[24] "The Ethernet: A Local Area Network: Data Link Layer and Physical Layer Specifications", Digital Equipment Corp., Intel, Xerox, Version 2.0, November 1982

[25] "IEEE Standard 802.3 - CSMA/CD Access Methods and Physical Layer Specification", IEEE Computer Society, 1985

[26] 82586 Local Area Network Coprocessor, data sheet, Intel

[27] M.Knirsch, Dissertation, TH Darmstadt, in Vorbereitung

[28] J.Pinkow, Diplomarbeit (1990), TH Darmstadt

[29] Timing Reference for pSOS$^+$/68020, Software Components Group

[30] A.Steinmetz, Diplomarbeit (1990), TH Darmstadt