Entwicklung einer Programmarchitektur für die zentrale Steuerung des S–DALINAC

und

Einbindung des magnetischen Strahlführungssystems der Experimentierhalle

Diplomarbeit

von

Dirk Schmischke

Institut für Kernphysik

Technische Hochschule Darmstadt

Juli 1993

Zusammenfassung

Das Ziel dieser Arbeit war es, die Bedienung des supraleitenden Elektronenbeschleunigers im Institut für Kernphysik an der TH Darmstadt S-DALINAC weiter als bisher zu vereinheitlichen. Zur Realisierung dieser Aufgabe wurden drei Teilaufgaben bewältigt.

Zum einen konnte die Bedienung der strahlführenden Elemente in der Experimentiersektion des Elektronenbeschleunigers in das Gesamtkonzept integriert werden.

Zum anderen sind nun die zur Überwachung der Funktion der Magnete wichtigen Shunt-Spannungen der Netzgeräte durch ein Analog-Digital-Konverter Modul meßbar. Diese Meßwerte wurden in die Datenstrukturen der Steuerungsprogramme integriert, so daß sie sich jederzeit auswerten lassen. Zu ihrer Kontrolle wurde ein Programm entwickelt, das sich an der grafischen Benutzeroberfläche der Beschleunigersteuerung orientiert.

Im Mittelpunkt dieser Arbeit stand jedoch die Entwicklung einer Kommunikations-Schnittstelle, die den Ablauf aller die Beschleunigersteuerung und –abfrage betreffenden Programme auf einem zentralen Rechner ermöglicht. Dabei wurde Wert auf einfachste Benutzbarkeit, größtmögliche Flexibilität und eine Architektur, die künftige Erweiterungen erleichtert gelegt. Realisiert wurde diese Schnittstelle durch ein Programmpaket, bestehend aus einem Server und einer leistungsfähigen Client-Funktionsbibliothek. Dieser zwischen Transportschicht und Anwenderprogrammen (Clients) eingefügte Server führt eine zusätzliche Abstraktionsebene ein, wodurch die Anwenderprogramme von der Kommunikation über das Netzwerk entkoppelt werden. Diese ist nun zentral im Server enthalten und läßt sich damit einfacher warten und weiter entwickeln.

Für Programm-Entwicklung und –test wurde zudem ein Test-Server entwickelt, der die gleiche Funktionalität, wie der Kommunikations-Server bietet, jedoch nicht in die Beschleunigersteuerung eingreifen kann. Damit ist es möglich, Steuerprogramme auch während des Strahlbetriebs zu testen, was die Programmierung wesentlich effektiver gestaltet und zu kürzeren Entwicklungszeiten führt.

Inhaltsverzeichnis

1 Einleitung **1**

2 Neue Programmarchitektur für eine zentrale Beschleunigersteuerung **3**

2.1 Konzeptionelle Überlegungen 3

2.2 Modifikation der Softwarearchitektur 4

2.3 Die Server-Architektur 5

2.3.1 Informationsbeschaffung und –weitergabe 5

2.3.2 Informationsfilter und –dienste 6

2.3.3 Applikations-Schnittstelle 7

2.3.4 Client–Server–Kommunikation 8

2.3.5 Mailboxen 10

2.4 Test-Server 12

3 Einbindung des 40°-Systems und der Extraktion **14**

3.1 Bisherige Ansteuerung der alten Strahlführung 14

3.2 Neues Konzept 14

3.2.1 Hardwaremodifikationen 15

3.2.2 Integration der seriell angesteuerten Netzgeräte 17

4 Anwendung der neuen Architektur zur Überwachung der Magnete **18**

4.1 Aufbau der Hardware 18

4.2 Integration in die Softwarearchitektur 19

5 Ergebnisse und Ausblick **21**

A Client-Server-Kommunikation **23**

A.1 Aufbau eines Datenpaketes 23

A.2 Geräte-Datenstruktur 26
A.3 Client-Funktionsbibliothek 27
A.4 Kommandotypen 30
A.5 Message-Typen 34

B ADC-Überwachungsprogramm **35**

C Erweiterungsplatine **37**

1 Einleitung

Seit mehr als 30 Jahren werden am Institut für Kernphysik der TH Darmstadt Elektronenstreuexperimente durchgeführt. Seit 1989 wird dazu der mit supraleitenden Beschleunigungsstrukturen arbeitende Linearbeschleuniger S-DALINAC [1] eingesetzt (vgl. Abb. 1.1). Im Gegensatz zu seinem Vorgänger DALINAC [2], der mit normalleitenden Beschleunigungsstrukturen ausgestattet war, ermöglicht er einen Dauerstrich– (continuous wave) Elektronenstrahl mit einem Tastverhältnis von 1 bei einer Maximalenergie von 130 MeV. Dadurch sind nun auch Koinzidenzexperimente möglich.

Durch die Vielzahl der zu steuernden Elemente, war der Einsatz von Rechnern unumgänglich. Im Rahmen einer Dissertation [3] und mehrerer Diplomarbeiten [4–9] wurde eine auf dem Netzwerk des Beschleunigerlabors basierende verteilte Beschleunigersteuerung entwickelt. Diese bestand aus einem Steuerrechner im Klystronraum und mehreren Rechnern, welche die Bedienung vom Kontrollraum aus ermöglichen.
Das Ziel dieser Arbeit war es nun, Steuerung und Bedienung des Beschleunigers noch weiter als bisher zu vereinheitlichen. Dazu mußten drei Aufgaben bewältigt werden:

- Entwurf und Realisierung einer neuen Programmarchitektur für eine zentrale Beschleunigersteuerung
- Integration der strahlführenden Elemente der Experimentiersektion in das Gesamtkonzept der Beschleunigersteuerung
- Überwachung der Magnetströme

In der vorliegenden Arbeit erfolgt daher zunächst eine Beschreibung dieser Teilschritte und anschließend eine Diskussion der erzielten Ergebnisse. Im Anhang findet sich schließlich eine detaillierte Beschreibung der erstellten Programmpakete und zugrundeliegenden Daten– und Kommunikationsstrukturen, sowie der Schaltplan der für die Ansteuerung der strahlführenden Elemente der Experimentiersektion zuständigen Erweiterungsplatine zu dem von [7] entwickelten Einplatinenrechner.

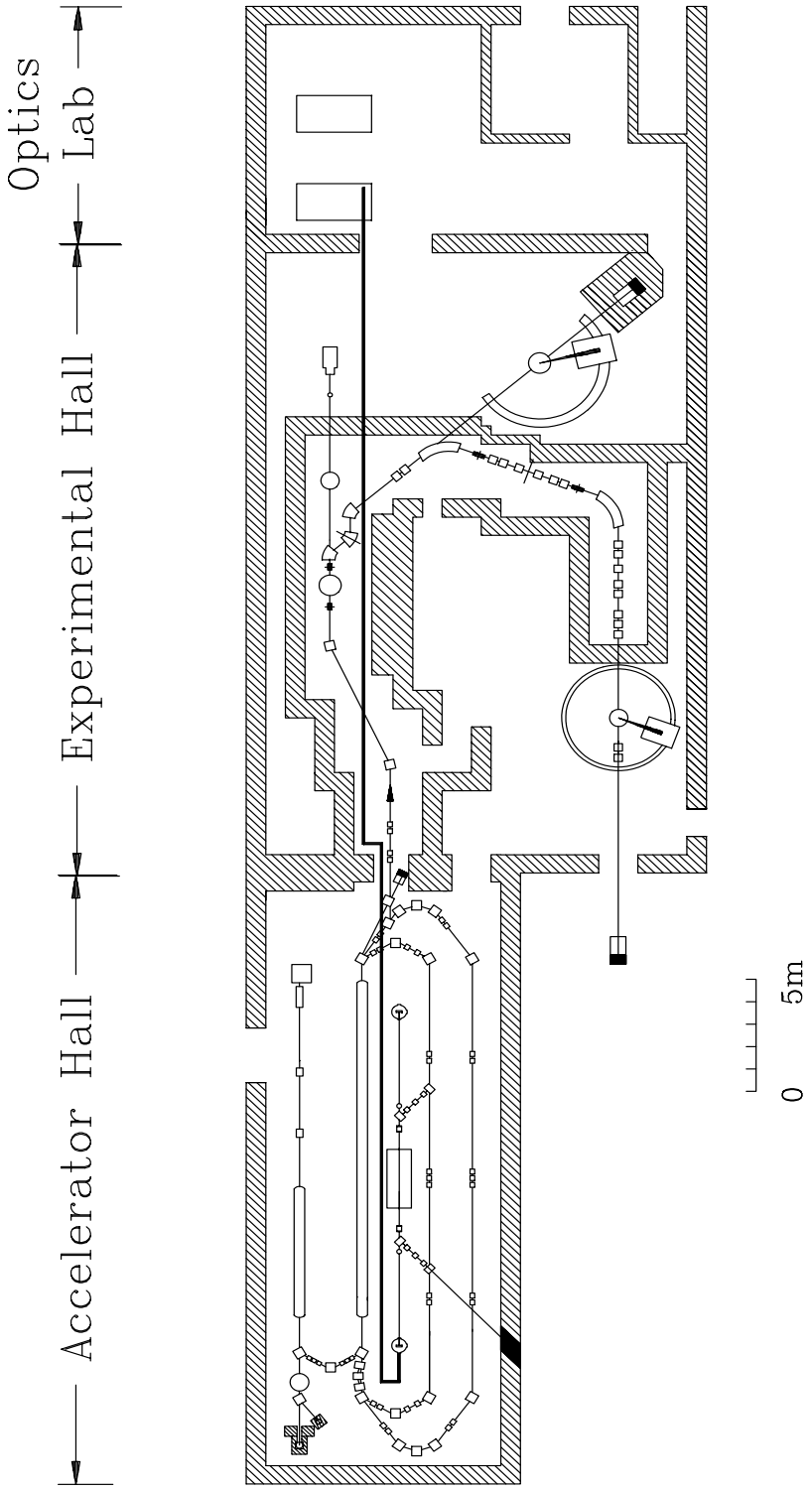

Abbildung 1.1: Grundriß des supraleitenden Beschleunigers S-DALINAC. Dargestellt sind auch die strahlführenden Dipol– und Quadrupolmagnete in der Experimentierhalle, deren Ansteuerung Gegenstand dieser Arbeit war.

2 Neue Programmarchitektur für eine zentrale Beschleunigersteuerung

Für Betrieb und Steuerung des S-DALINAC wird eine verteilte Rechnersteuerung benutzt. Diese ist ausführlich in [3] beschrieben. Während es in der ersten Ausbauphase darauf ankam, daß die wichtigsten Funktionen zum Betrieb des Beschleunigers vorhanden waren, sollten nun vorhandene Funktionen besser integriert und zusätzliche Überwachungsfunktionen implementiert werden.
Im folgenden Kapitel wird auf das neue Konzept einer zentralen Beschleunigersteuerung und –überwachung auf einem für diesen Zweck reservierten Rechner eingegangen.

2.1 Konzeptionelle Überlegungen

Die Rechnersteuerung des S-DALINAC teilt sich in zwei Bereiche auf: Lesen und Setzen aller Ist– und Sollwerte und die Bedienoberfläche.
Beide Bereiche werden durch das im Beschleunigerlabor eingesetzte Netzwerk auf Ethernet-Basis verbunden. Den Bereich Steuerung übernimmt hierbei der Steuerrechner RECYC. Er ist für die Ansteuerung der Magnet-Netzgeräte, Kameras, Targetsteuerungen und Faraday-Cups zuständig. Die Bedienung erfolgt entweder lokal an einem direkt mit RECYC verbundenen Terminal oder über ein spezielles Ethernet-Protokoll auf einem anderen Rechner des Netzwerks.
Zur Zeit wird das zur Ethernetverbindung von Steuer- und Bedienrechner (LCP — Linac Control Protocol) definierte Protokoll [5] von zwei Programmen benutzt: von MCP (Master Control Program) , welches die Einstellung der Magnete ermöglicht [4] und von VIEW [9], das für die Steuerung der Kameras, Targets und Faraday-Cups zuständig ist. Im Rahmen dieser Diplomarbeit wurde ADC_MCP entwickelt, mit dem die einwandfreie Funktion der Magnete durch Kontrolle der tatsächlich fließenden Magnetstöme überwacht werden kann. Vor der Inbetriebnahme befindet sich ein Programmsystem zur Verwaltung aller Einstellparameter auf der Basis einer relationalen Datenbank [10]. Dieses und das Programm XBEAM [11], eine Simulation der Strahlführung, müssen ebenso integriert werden, da beide die über das Netzwerk von RECYC bereitgestellten Beschleunigerparameter benötigen.
Wegen der Struktur der Netzwerksoftware im Betriebssystem ist es jedoch nicht möglich, mehr als ein Programm pro Rechner ablaufen zu lassen, das den LCP-Transport benutzt. Mit der bisherigen Lösung würde dies die Bedienung bereits auf fünf Rechner verteilen. Damit wäre die Bedienung des Beschleunigers abhängig von der Belastung jedes Rechners durch andere auf ihm laufende Programme, die nicht für die Steuerung des Beschleuniger zuständig sind. Hierdurch kann es zu störenden Verzögerungen in der Bedienung kommen.

Desweiteren würde die Zahl der übertragenen Ethernet-Datenpakete mit der Anzahl der das LCP benutzenden Programme ansteigen und somit das Rechnernetz unnötig belasten.
Das Ziel dieser Arbeit war es nun, eine Methode zu entwickeln, mit der es möglich ist, alle die Beschleunigersteuerung betreffenden Programme auf einem diesem Zweck vorbehaltenen Rechner zu implementieren.

2.2 Modifikation der Softwarearchitektur

Wie bereits eingangs erwähnt, beruhte die bisherige Rechnersteuerung des Beschleunigers auf einer Zweiteilung der Aufgaben in einen Steuerungs- und einen Bedienungsteil. Die Schnittstelle zwischen diesen beiden Programmpaketen war das LCP, und die Verbindung erfolgte über das Ethernet. Das LCP erlaubt Single- und Multicast-Empfängeradressen, das heißt ein über das Ethernet geschicktes Datenpaket kann von einem oder mehreren Rechnern empfangen werden.
Schickt der Steuerrechner RECYC Daten an mehrere Empfänger über den gleichen Ethernet-Protokolltyp, so ergibt sich das Problem, daß verschiedene Eingabebefehle für jeden Empfänger aufgesetzt werden müßten. Zwar können mehrere QIOs[1] [13] für einen Protokolltyp auf einem Rechner aktiv sein, jedoch gilt hier, daß der zuerst gestartete QIO das erste Datenpaket empfängt, der zweite das nächste Datenpaket und so weiter. Das verhindert den simultanen Ablauf von Programmen auf einem Rechner, die das gleiche Protokoll (LCP) benutzen. Der Ausweg aus diesem Dilemma bestand bisher darin, jedes das LCP benutzende Programm auf einem eigenen Rechner auszuführen.
Die Nachteile dieser Methode sind die Dezentralisierung der Beschleunigersteuerung, die Abhängigkeit von der Rechnerbelastung und die Vielzahl der nötigen Korrekturen bei Veränderung der Kommunikationsstruktur. Die Dezentralisierung hat aber vor allem in der Programmentwicklung entscheidende Nachteile: Für die Synchronisation der QIOs werden sogenannte AST-Routinen (Asynchronous System Trap, vgl. Kapitel 2.3.5) verwendet [14]. Bei Entwicklung und Test dieser sehr systemnahen Programmteile können die Programme in einen nicht auflösbaren Systemzustand RWAST (Resource Wait in Asynchronous System Trap) gehen, falls sich in ihnen nicht vollständig aufgesetzte Kommunikationskanäle (d.h. solche ohne Gegenstück) befinden [12]. Um die Programme neu zu aktivieren, ist ein Neustart des Systems (Reboot) erforderlich. Dies verbietet sich jedoch, da die Rechner von vielen Benutzern belegt sind. Sinnvoller ist in diesem Zusammenhang die Reservierung eines Rechners nur für die Beschleunigersteuerung, auf dem sämtliche damit verbundenen Programme laufen können.
Einen weitereren Nachteil der bisherigen Kommunikationsschnittstelle war, daß hardwarenahe Programmteile und die Abhängigkeit von der Struktur des Protokolls LCP in jedem Anwendungsprogramm vorhanden war. Veränderungen an

[1] QIO: Queued Input Ouput — Ein– Ausgabebefehl des Betriebssystems

dieser Struktur zogen Korrekturen in allen diese Schnittstelle benutzenden Programmen nach sich, da die entsprechenden Programmodule für die Ethernet-Kommunikation in allen genannten Anwendungsprogrammen eingebunden sind. Besser wäre es, eine hardware– und protokollunabhängige Schnittstelle einzusetzen. Dies kann mit dem modernen Konzept einer *Client-Server-Architektur* erreicht werden, das wegen seiner Flexibilität in vielen kommerziellen Applikationen[2] gebräuchlich ist:

> Ein Server-Programm wird zwischen die Kommunikationschnittstelle und die Anwendungsprogramme (Clients) gesetzt, versorgt alle Client-Programme mit Daten und stellt Datendienste zur Verfügung. Dies schafft eine zusätzliche Kommunikationsschicht, die die Programme unabhängig von Änderungen an Hardware und Kommunikationsprotokollen macht und es ermöglicht, den jeweiligen Client-Programmen nur die Informationen zu geben, die diese angefordert haben, was den Kommunikationsaufwand reduziert und die Programmwartung erleichtert.

Im Hinblick auf eine mögliche Ersetzung des alten Beschleuniger-Steuerungsrechners (RECYC), einer PDP 11/73, durch eine VAX unter VMS oder VAXELN und des damit notwendigen Neuentwurfs der Beschleunigersteuerung wäre dies von Vorteil: Programme, welche die Server-Schnittstelle benutzen, müssen bei einem Wechsel des RECYC nicht umgeschrieben werden, lediglich der Server muß an die neuen Protokolle angepaßt werden. Der Server unterstützt bereits jetzt alle Funktionalitäten, die eine Modernisierung der Beschleunigersteuerung beinhalten sollte.

2.3 Die Server-Architektur

In den folgenden fünf Kapiteln wird darauf eingegangen, wie der Server aufgebaut ist, welche Funktionen er erfüllt und wie er implementiert ist. Den Abschluß bildet, da dies für diesen Server von zentraler Bedeutung ist, eine Erklärung der Kommunikation mittels Mailboxen [15].

2.3.1 Informationsbeschaffung und –weitergabe

Um dem Server eine größtmögliche Flexibilität bezüglich Erweiterungen und Änderungen zu geben, sind Informationsbeschaffung, also hier der Zugriff auf die Systemparameter des Beschleunigers über eine Ethernet-Verbindung (s. Abb. 2.1) und deren Weitergabe an Client-Programme strikt getrennt. Die Ethernet-Verbindung wird durch ein Programm (TRANCEIVE [5]) hergestellt, welches das

[2] Man denke nur an die X-Windows-Server auf dem Workstation-Markt, die DDE– und OLE-Server in Microsoft Windows oder Datenbank-Server wie IDAPI von Borland.

LCP zur Kommunikation mit RECYC benutzt und nur zur reinen Datenübertragung über das Netz dient. Als Bindeglied zwischen Informationsbeschaffung und –weitergabe fungieren gemeinsame Datenstrukturen und Ereignis-Flags. In den Datenstrukturen werden die von der Informationsbeschaffung gesammelten Daten abgelegt und können dort für die Informationsweitergabe an die Client-Prozesse benutzt werden. Zur Erkennung geänderter Daten dienen Ereignis-Flags. Die Informationsweitergabe an die angeschlossenen Client-Prozesse erfolgt im Server durch eine periodisch über einen Timer aufgerufene Service-Routine, die die ankommenden Informationen in geeigneter Form direkt an die Zielprozesse weitergibt und damit beide Teile entkoppelt.
Das bietet zwei Vorteile. Zum einen wird der Rechner entlastet. Besonders in Fällen, wo eine schnelle Folge von Datenpaketen z.B. die Veränderung nur eines Parameters betrifft, ist es nicht nötig, jedes Datenpaket weiterzuleiten. Hier genügt es, die ankommenden Datenpakete in der Zeit zwischen den Aktivierungen der Service-Routine zu sammeln und nur die jeweils aktuellsten weiterzugeben. Die Verzögerung zwischen ankommenden Daten und deren Weiterleitung läßt sich durch eine Verkürzung der Dauer zwischen den Aufrufen der Service-Routine soweit verringern, daß keine Beeinträchtigung des Echtzeitverhaltens mehr bemerkt werden kann[3]. Momentan liegt diese Zeit bei 100 ms. Zum Vergleich: bei MCP, dem Programm zur Steuerung der Magnete, wird ein Auffrischen der angezeigten Daten alle 400 ms durchgeführt.
Zum anderen läßt sich diese Struktur wesentlich einfacher ändern und erweitern als eine mit direkter Verbindung zwischen Informationsbeschaffung und –weitergabe.
Wie flexibel diese Struktur ist, zeigte sich bei der Programmierung des Test-Servers. Hierbei wurde der gesamte Kommunikationsteil (Informationsbeschaffung) durch eine Simulation ersetzt. Dieser Test-Server konnte an einem Nachmittag aus dem bestehenden Server erzeugt werden.

2.3.2 Informationsfilter und –dienste

Nicht jeder Client benötigt alle Informationen, die der Server zur Verfügung stellen kann. Aus diesem Grund und wegen der Reduktion der zu übertragenden Datenmenge bestimmt jeder Client, welche Informationen er erhalten möchte. Dazu trägt er sich in verschiedene Informationslisten, das sind verkettete Listen im Server, die bestimmen, über welche Ereignisse ein Client unterrichtet werden soll, ein. Empfängt der Server ein bestimmtes Ereignis über die Ethernetschnittstelle oder von einem Client, so wird ein Ereignis-Flag gesetzt, das den Informationsweitergabeteil des Servers über neue Ereignisse informiert.
In der Service-Routine wird schließlich in den Listen nachgesehen, welcher Client über dieses Ereignis unterrichtet werden soll, und die entsprechende Nachricht

[3]Echtzeit bedeutet im Falle einer menschlichen Bedienung: innerhalb 1/16 Sekunde, der zeitlichen Wahrnehmungsgrenze.

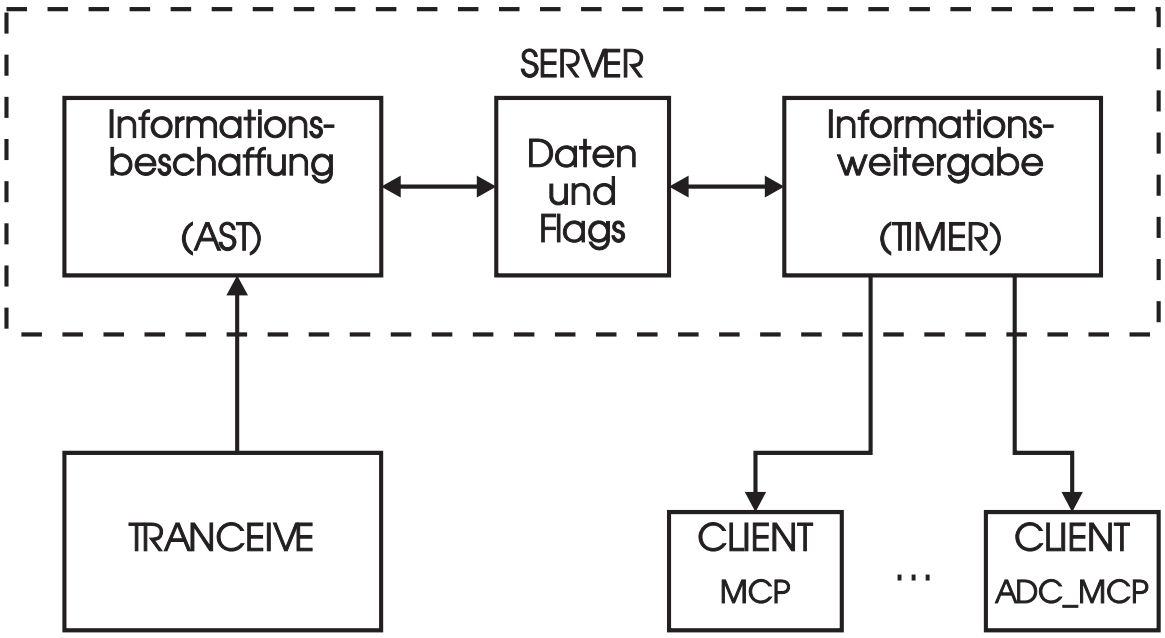

Abbildung 2.1: Interner Aufbau des Servers. Man erkennt die Trennung in zwei funktionale Bereiche, die über einen gemeinsamer Daten– und Flag-Bereich miteinander kommunizieren.

wird in die zu diesem Client gehörende Mailbox geschrieben. Die Benutzung dieser in VMS gebräuchlichen Kommunikationmöglichkeit zwischen Prozessen ist in Kapitel 2.3.5 näher beschrieben.

2.3.3 Applikations-Schnittstelle

Der Entwickler der für die Beschleunigerbedienung zuständigen Programme sollte die logischen Namen der Geräte, die zur Beschreibung des Beschleunigers in Gebrauch sind, verwenden können. Das heißt, das Anwendungsprogramm auf dem Bedienungsrechner braucht die Namensstruktur, die auf dem Steuerrechner in Anlehnung an das Betriebssystem RSX implementiert ist, nicht zu kennen. So ist es nicht von Bedeutung, daß der Magnet E1BM01 über Treiber NJ, Device 01, Unit 01 angesteuert wird. Wichtig ist nur, diesen Magneten unter seinem Namen ansprechen zu können. Bisher geschah die Übersetzung Name in Treiber/Device/Unit-Nummer[3] durch mehrere Tabellen, eine auf RECYC-Seite und eine pro Bedienungsprogramm. Die Übertragung zwischen den Rechnern über das LCP benutzt zur Zeit noch diese Treiber/Device/Unit-Nummer Konvention. Der Server, der die Namen auf diese Strukturen abbildet, bietet jetzt diesen abstrakten Zugriff auf Geräte an und ist damit einfacher zu benutzen als die bisherige Lösung, bei der diese Umsetzung im Anwendungsprogramm erfolgen mußte.

Wird in Zukunft die Hardware erweitert, umgruppiert oder wird das Übertragungsprotokoll geändert, bemerkt der Benutzer davon nichts, da diese für ihn unwichtigen Informationen vor ihm verborgen werden. Eine Adaption der Client-Programme erübrigt sich also, was eine erhebliche Arbeitserleichterung bedeutet und mögliche Fehlerquellen einschränkt.
Die Erstellung neuer Anwendungen wird durch die zur Verfügung gestellte Funktionsbibliothek, die bereits die wichtigsten Funktionen zur Kommunikation zwischen Client und Server in sehr komfortabler Form enthält, wesentlich vereinfacht. Der Benutzer braucht sich bei diesen Funktionen nicht mehr um die Details des Protokolls, wie Multiset-Kommandos, variable Datenstruktur und optimale Ausnutzung der Mailbox-Datenblockbreite zu kümmern. Zu diesen Funktionen gehört:

- der Aufbau einer Geräteliste[4] durch den Server,
- eine Lese-Funktion, welche die übertragenen Daten auswertet und an die richtigen Stellen in der Geräteliste setzt,
- eine Set-Funktion, die eine komplette Geräteliste an den Server übertragen kann
- und zwei Umrechnungsfunktionen, die zwischen Skalenteilen und magnetischen Feldstärken beziehungsweise magnetischen Feldgradienten bei Quadrupolmagneten konvertieren können.

2.3.4 Client–Server–Kommunikation

Die Kommunikation zwischen Server und Client verläuft über zwei Mailboxen. Eine Mailbox ist für die von den Clients gesendeten Kommandos zuständig, die andere überträgt Datenpakete vom Server zum Client.
Es gibt nur eine zentrale Kommando-Mailbox zum Server, durch die Befehle aller Clients laufen. Anders sieht es bei den Daten-Mailboxen aus, durch die der Server Datenpakete an einen Client verschickt. Die ursprüngliche Idee war hier, eine Mailbox für alle Client-Programme zu benutzen und die Kommunikation in Art eines Token-Ring-Protokolls auszuführen (s. Abb. 2.2). Dies wurde aus folgendem Grund fallengelassen:
Falls sämtliche Clients ihre Daten aus genau einer Mailbox beziehen würden, bedeutet dies aber auch, daß jeder Client einen lesenden QIO inklusive AST-Routine auf ein und dieselbe Mailbox geöffnet hätte. QIOs auf Mailboxen funktionieren auf die gleiche Art wie andere QIOs, deren ASTs in Warteschlangen abgearbeitet werden. Das heißt jedoch, daß nur die älteste AST-Routine das Datenpaket

[4]Die Geräteparameter sind in Form einer verketteten Liste gespeichert. Das bietet den Vorteil, die Anzahl der Geräte nicht schon bei der Programmerstellung festlegen zu müssen.

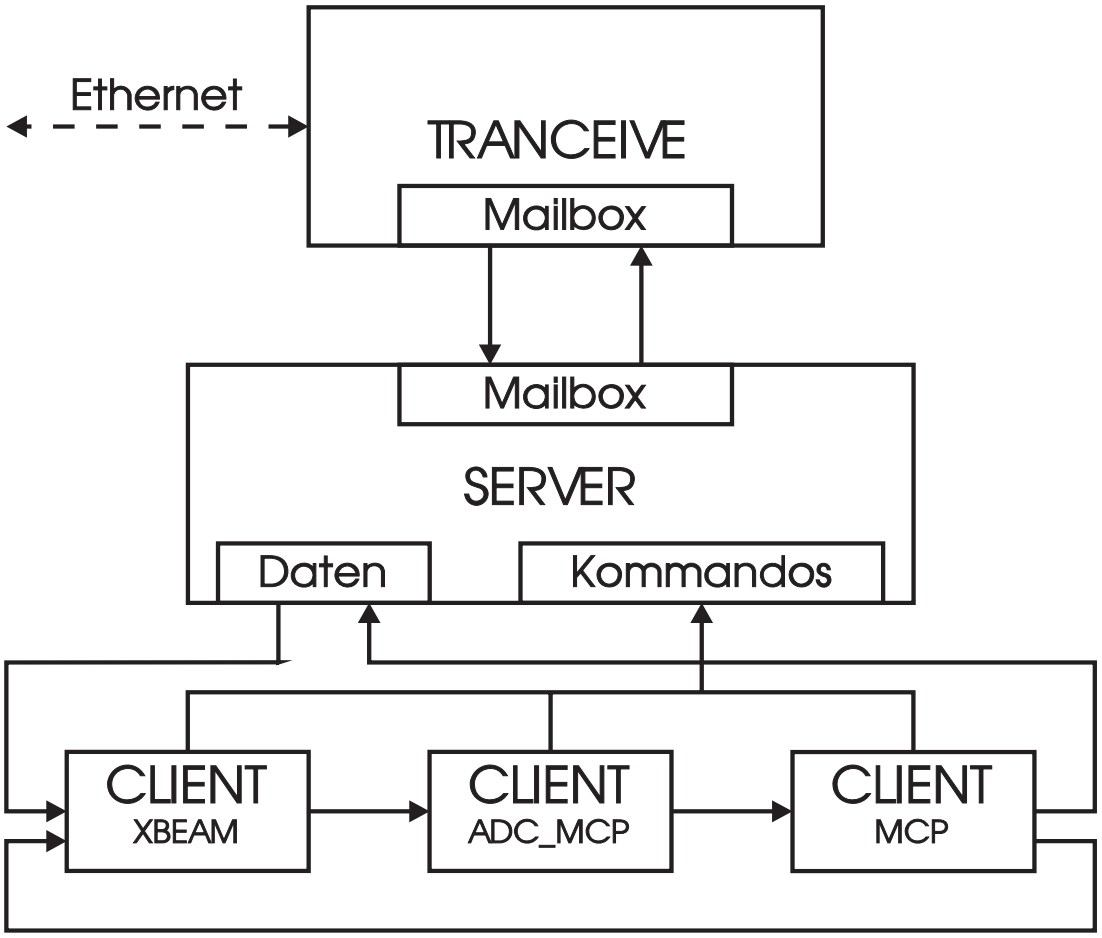

Abbildung 2.2: Beispiel einer Token-Ring-Struktur. Daten werden von Client zu Client weitergeleitet, bis sie ihr Ziel erreicht haben.

bekäme, die nachfolgenden AST-Routinen müßten auf die nächsten Datenpakete warten. Hierdurch kämen die Datenpakete nicht immer an die richtige Stelle.
Um sie dennoch zum richtigen Empfänger zu transportieren, müßten die Client-AST-Routinen prüfen, ob das Datenpaket für sie bestimmt war, indem sie ihre Adresse mit der Ziel-Adresse vergleichen. Stimmen beide überein, würden die Daten weiterverarbeitet, im anderen Fall erneut in die Mailbox geschrieben und dort von der nächsten aktiven AST-Routine bearbeitet werden. Auf diese Weise würden die Datenpakete solange von Client zu Client weitergereicht werden, bis sie am Ziel wären. Würde die Kommunikation jedoch an einer Stelle gestört werden, könnten Daten verloren gehen oder der ganze Ring könnte zum Stillstand kommen. Zudem würde das Weiterreichen der Datenpakete unnötig Zeit kosten.
Aus diesem Grund fiel die Wahl auf eine exklusive Datenmailbox zwischen Server und jedem Client (s. Abb. 2.3). Kommt es nun zu einem Ausfall eines Clients,

werden die anderen dadurch nicht beeinträchtigt. Nach dem Start eines Clients erzeugt dieser eine Mailbox, deren Name seine PID (Process IDentifikation number) enthält. Diese PID wird später implizit bei jedem Kommando, das der Client an den Server schickt, mit übertragen und kann im Server aus dem Status-Block des QIOs ausgelesen werden. Damit weiß der Server immer, welcher Client mit ihm kommuniziert und in welche Mailbox er seine Antworten schicken soll.
Das Erzeugen einer Mailbox reicht allerdings nicht aus, um eine Kommunikation zwischen Client und Server aufzubauen, denn der Server hat keine direkte Möglichkeit, dies zu registrieren. Aus diesem Grund schickt der Client ein „open-communication"-Kommando an den Server und wartet auf eine Bestätigung, die ihm mitteilt, daß er registriert wurde. Die komplette Beschreibung des Kommunikationsprotokolls über die Mailboxen mit allen Kommandos und Datenstrukturen findet sich im Anhang.
Einen Spezialfall gilt es bei der Kommunikation gesondert zu behandeln. Durch einen Programmfehler oder unvorhergesehene Umstände kann es vorkommen, daß sich ein Client vor seinem Ende nicht ordnungsgemäß bei dem Server abmeldet und dieser weiterhin Datenpakete an diesen Client schickt. Da das Ziel nicht mehr existiert, füllt sich die Mailbox bis zur bei der Erzeugung festgelegten Grenze und blockiert so wertvolle Systemressourcen.
Zur Erkennung solcher nicht mehr aktiven Clients läuft in jedem Clientprogramm ein Timer mit, der in periodischen Abständen eine Aktivitäts–Meldung an den Server verschickt. Sollte diese Meldung im Server eine bestimmte Zeit ausbleiben, wird der entsprechende Client als inaktiv markiert und aus allen Informationslisten ausgetragen; seine Mailbox wird gelöscht.

2.3.5 Mailboxen

Die Kommunikation über Mailboxen ist für Interprozesskommunikation von entscheidender Bedeutung, deshalb folgt nun eine kurze Einführung.
Es gibt zwei Klassen von Mailboxen, permanente und temporäre. Permanente Mailboxen existieren unabhängig von auf sie geöffneten Kommunikations-Kanälen und sind systemweit erreichbar. Temporäre Mailboxen dagegen sind nur solange vorhanden, wie Kommunikations-Kanäle auf sie geöffnet sind. Sie sind entweder prozeßweit oder gruppenweit ansprechbar. Aus Sicherheitsgründen sollte es nur den explizit für die Beschleunigersteuerung zuständigen Programmen, die eine Gruppe bilden, gestattet sein, Einstellungen vorzunehmen. Somit fiel die Wahl auf gruppenweit definierte temporäre Mailboxen als Kommunikationsmedium.
Mailboxen sind keine physikalisch existenten Geräte wie Festplatten oder Drucker. Dennoch lassen sie sich durch die Standard Ein/Ausgabebefehle (hier bevorzugt durch QIO) ansprechen. Daten können in sie hineingeschrieben und aus ihnen herausgelesen werden. Wie bei anderen Geräten lassen sich auch bei Mailboxen QIOs mit einer AST-Routine verbinden, mit denen man die Ein– Ausgabevorgänge auf einfache Weise synchronisieren kann, was in Verbindung mit der Tatsache, daß

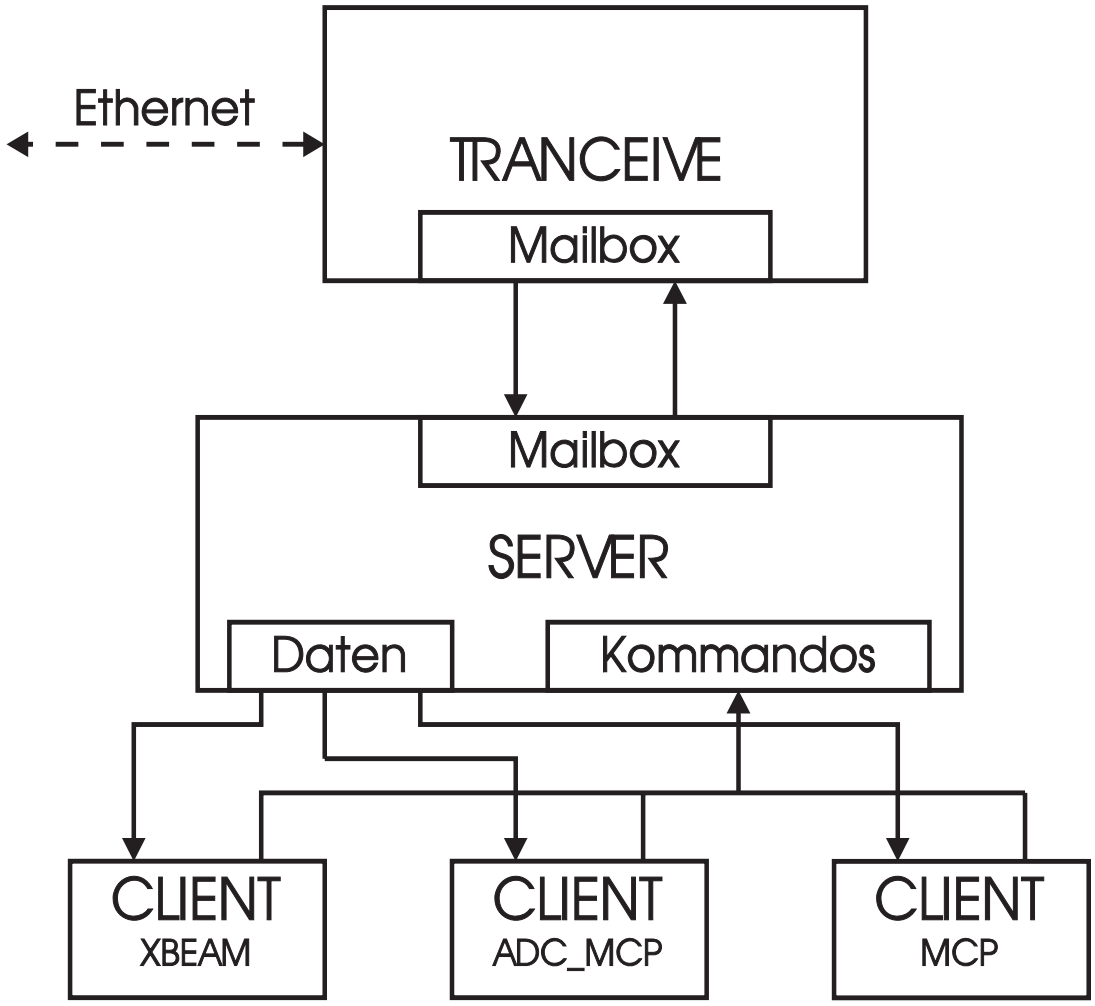

Abbildung 2.3: Realisierte Client-Server-Struktur. Die Anbindung an den Steuerrechner erfolgt über das Ethernet durch den Prozess TRANCEIVE. Dieser ist durch zwei Mailboxen mit dem Server verbunden. Die Kommunikation zwischen Clients und Server erfolgt über getrennte Datenmailboxen und eine gemeinsame Kommandomailbox.

sie im Hauptspeicher des Rechners residieren, die Kommunikation äußerst schnell macht.

QIOs sind Ein/Ausgabebefehle, die in eine Warteschlange (Queue) eingereiht werden und dort auf die Erfüllung ihres Auftrages warten. Jeder QIO kann optional mit einem AST verbunden werden. Der AST aktiviert bei Auftreten eines bestimmten, bezüglich des normalen Programmablaufs asynchronen Ereignisses (also beispielsweise ankommende Daten oder ein beendeter Ausgabebefehl) eine AST-Routine, welche die weitere Bearbeitung durchführt. Die Aktivierung dieser

AST-Routine erfolgt asynchron zum normalen Programmablauf und ist weitgehend unabhängig von der Zuteilung der Rechenzeit an die einzelnen Programme durch das Betriebssystem. Dies führt zu sehr kurzen Reaktionszeiten, eine wichtige Vorbedingung für ein echtzeitfähiges Multiprogramm-System.
Es gibt noch andere Möglichkeiten der Prozeßkommunikation, wie globale Datensegmente oder globale Event-Flags, die schneller ablaufen. Diese bieten jedoch nicht den Vorteil der völligen Abkapselung der Client-Programme, da alle Anwendungsprogramme an zentrale Datenbereiche (shared Images) gebunden werden müssen, so daß diese Verfahren hier nicht benutzt wurden.

2.4 Test-Server

Bei der Programmerstellung erwies es sich bisher als äußerst hinderlich, daß während des Beschleunigerbetriebs keine Benutzung des LCP möglich war. Fehlerhafte Daten hätten hier zu großem Schaden führen können und den regulären Strahlbetrieb erheblich gestört.

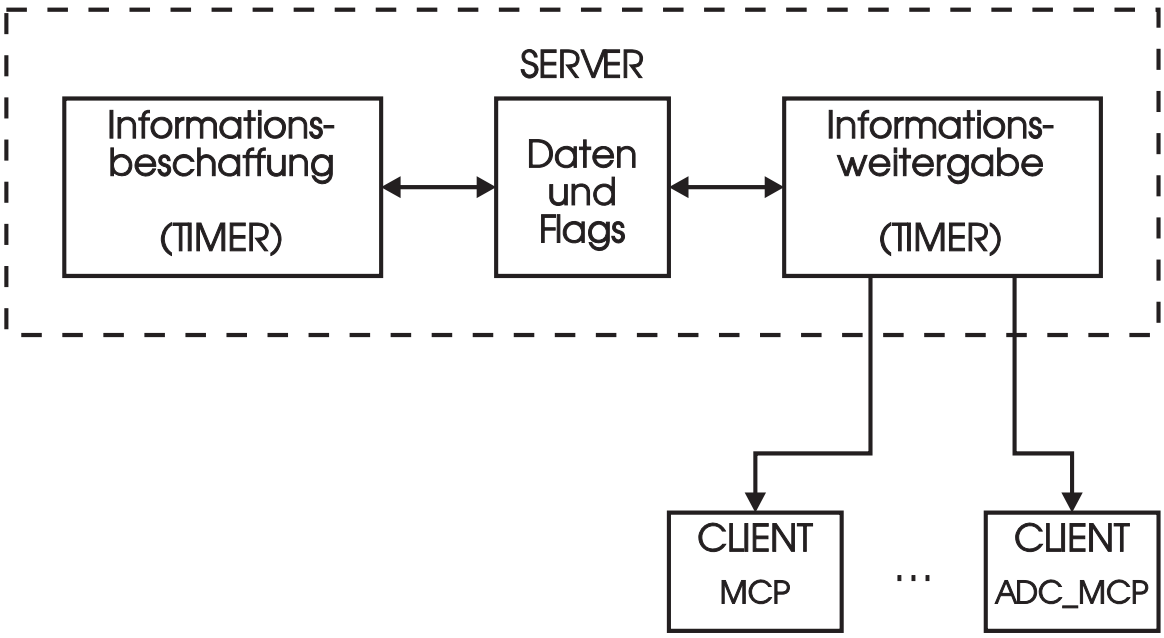

Abbildung 2.4: Interner Aufbau des Test-Servers.

Um dieses Problem zu umgehen, wurde ein Test-Server entwickelt, der die gleiche Funktionalität wie der eigentliche Server bietet, jedoch keine Beschleunigereinstellungen vornehmen kann. Dazu wurde seine Informationsbeschaffungs-Methode geändert. Der Test-Server bekommt seine Daten nicht von dem Subprozess TRANCEIVE über die Ethernetkommunikation (LCP) mit dem Steuerrechner RECYC, sondern verändert periodisch eine Anzahl von Daten auf zufällige

Werte und setzt die entsprechenden Ereignis-Flags. Die Informationsweitergabe-Methode wurde nicht geändert, sodaß für Client-Programme kein Unterschied zu dem richtigen Server besteht. Natürlich müßen dieser Server und die ihn benutzenden Client-Programme auf einem anderen Rechner laufen, als dem für die Beschleunigersteuerung zuständigen, da sie sich sonst gegenseitig stören würden. Client-Programme können nun während des Beschleunigerbetriebes entwickelt und getestet werden. Nach erfolgreichem Test können sie dann einfach auf dem Beschleunigersteuerungs-Rechner in Betrieb genommen werden.

3 Einbindung des 40°-Systems und der Extraktion

Im Folgenden wird auf das Konzept der einheitlichen und zentralen Ansteuerung der Magnetnetzteile durch den Steuerrechner RECYC eingegangen. Dabei wird beschrieben, wie die Ansteuerung der Magnete der alten Strahlführung zu Beginn dieser Diplomarbeit realisiert war und wie sie im Rahmen dieser Arbeit in die zentrale Beschleunigersteuerung integriert wurde.

3.1 Bisherige Ansteuerung der alten Strahlführung

Zur Zeit werden drei verschiedene Netzgeräteklassen zur Ansteuerung der Magnete des S-DALINAC benutzt:

1. direkt mit dem Steuerrechner RECYC verbundene Netzgeräte zur Ansteuerung der Magnete des Recyclotrons,
2. über eine serielle Leitung ansteuerbare Netzgeräte der alten Strahlführung,
3. ein ebenfalls seriell ansteuerbares Netzgerät zur Versorgung der Magnete des **40**°-Systems.

Die erste Netzgeräteklasse war bereits zu Beginn dieser Diplomarbeit in die Rechnersteuerung des S-DALINAC integriert und ließ sich vom Kontrollraum aus steuern. Die anderen beiden Netzgeräteklassen wurden im Rahmen dieser Arbeit eingebunden.
Zur Ansteuerung der Magnete (Punkt 2) die aufgrund ihrer günstigen Anordnung und ihrer Funktionstüchtigkeit in den Aufbau des S-DALINAC übernommenen worden sind, wurde ein Rechner-System [7] eingesetzt. Dieses bestand aus 13 Einplatinenrechnern, die über eine serielle RS-232-Schnittstelle angesteuert werden konnten. Jede dieser Platinen war für ein Magnetnetzteil zuständig und erlaubte das Setzen und Auslesen der Steuerspannungen der Netzteile. Die Bedienung erfolgte über ein Terminal und war nicht in die Rechnersteuerung des S-DALINAC integriert.

3.2 Neues Konzept

Das Ziel dieser Arbeit war es:

1. den Betrieb der Einplatinenrechner durch Hardwaremodifikation störsicher zu machen,
2. die Ansteuerung der Magnete der alten Strahlführung in die Rechnersteuerung des S-DALINAC zu integrieren und
3. die Ansteuerung des **40**°-Systems ebenfalls in die Steuerung zu integrieren.

3.2.1 Hardwaremodifikationen

Das Konzept, 13 Einplatinen-Rechner zur Ansteuerung der 13 Magnetnetzteile der alten Strahlführung zu benutzen, erwies sich bei Stromausfällen als ungünstig. So waren z.B. nach einem Störfall 8 der 13 Rechner defekt. Der Grund war, daß Leitungstreiber und –empfänger der RS-232-Schnittstelle nicht ausreichend gegen Störspannungen gesichert waren, durchbrannten und noch einige nachfolgende Schaltkreise zerstörten. Die große Zahl der Einplatinen-Rechner ist ein weiterer Unsicherheitsfaktor, insbesondere da die Aufgabe der Ansteuerung der Magnetnetzteile von der Rechenleistung her auch von einem Rechner bewältigt werden kann. Die zusätzlich eingesetzten Rechner erhöhen in diesem Fall nur die Ausfallwahrscheinlichkeit.
Aus diesem Störfall lassen sich zwei Forderungen an ein sichereres Konzept der Magnet-Ansteuerung durch die verwendeten Einplatinencomputer ableiten:

1. Verbesserter Schutz der RS-232-Schnittstelle.

2. Verwendung eines Einplatinen-Rechners für alle Magnetnetzgeräte.

Die erste Forderung ließ sich einfach erfüllen:
Der in der Schnittstelle eingesetzte Maxim MAX-241-Treiberbaustein ist gegen Störspannungen bis zu $\pm$ 30 Volt geschützt [18]. Zur Erhöhung der Störspannungsfestigkeit wurden Kontaktschutzdioden eingebaut. Diese begrenzen die auftretenden Spannungen auf $\pm$ 22 Volt und das selbst bei sehr hohem, kurzzeitigem Stromfluß ($\leq$ 100 A, $\leq$ 10 ms). Dies hat bisher weitere Ausfälle verhindert.
Zur Erfüllung der zweiten Forderung mußte eine Erweiterungsplatine entwickelt werden, da die vorhandenen Ausgabeleitungen eines Einplatinen-Rechners nicht ausreichten. Diese mußte einen Multiplexer aufnehmen, der den einen auf dem Einplatinenrechner vorhandenen Ausgabekanal auf die vielen durch den Wegfall der übrigen Rechner benötigten Kanäle umsetzte. Damit diese in das vorgesehene Gehäuse, einen NIM-Rahmen, paßte, wurde die Schaltung auf drei Platinen verteilt.
Entworfen wurde sie mit dem Platinenerstellungsprogramm Ranger 3. Die automatische Leiterbahnentflechtung (Autorouter) erwies sich in diesem Falle als unbrauchbar, da die Platinen im Institut hergestellt werden sollten und deshalb Wert auf eine möglichst geringe Anzahl von Durchkontaktierungen (Vias) gelegt werden musste. Von 400 Vias, die der Autorouter zur Lösung der Aufgabe benötigte, konnte durch manuelles Entflechten auf acht Vias optimiert werden.
Die für die Ansteuerung der Magnetnetzteile zuständigen Digital-Analog-Konverter (DAC) werden über eine parallele Schnittstelle auf den Einplatinencomputern angesteuert [7]. Diese Schnittstelle besteht aus drei 8-Bit-Latches und bietet somit 24 Ausgabeleitungen, von denen 20 für die Ansteuerung des DACs benötigt werden.
Vier Ausgabeleitungen waren nicht beschaltet und konnten auf der Erweiterungsplatine zur Selektion von 15 verschiedenen Latch-Gruppen benutzt werden. Für

jeden DAC wird eine Latchgruppe benötigt, da die Daten permanent anliegen müssen [7]. Sind alle vier Ausgabeleitungen auf Low-Pegel, wird keine der 15 Gruppen angesprochen. Dies ist der Grundzustand, in den nach jedem Ansprechen einer Gruppe zurückgekehrt werden muß, damit die Information in den Latches gespeichert bleibt. Die Auswahl der Latch-Gruppen geschieht auf jeder Erweiterungsplatine, die fünf Latch-Gruppen enthält, durch einen $4 \rightarrow 16$ Dekoder (74154), von dessen 16 Ausgängen fünf über ein Steckfeld, mit den Selektionsleitungen der Latch-Gruppen verbunden werden können.
In Tabelle 3.1 findet man eine Auflistung aller von der Erweiterungsplatine bedienten Geräte und ihrer logischen Adressen in der Einplatinenrechner-Software, die aufgrund der veränderten Hardware neu entwickelt werden mußte.

Ein erster Versuch, die Verbindung über eine RS-232-Leitung herzustellen, führte zu fehlerhaften Ergebnissen. Die Verbindung zwischen RECYC und den beiden Systemen mußte wegen der großen Entfernung (ca. 30 Meter) und der starken Störfelder im Betrieb (verursacht durch 50 Hz-Einstreuungen und Schalt-Spannungsspitzen) durch galvanisch getrennte, differentielle stromgesteuerte Leitungstreiber erfolgen. Diese Leitungstreiber waren bereits früher am Institut gebaut worden. Sie wurden vor der Einführung der Terminalserver zur störungsfreien Übertragung der Daten zwischen den einzelnen Terminals und den seriellen Ports der Rechner benutzt.
Ein positiver Nebeneffekt der Zwischenschaltung der differentiellen Leitungstrei-

Gerätename	Alte Bezeichnung	Logische Adresse
E2QR03	R3	1
E2BM02	70 Grad	2
E2QR02	R2/R4	3
E2QR01	R1/R5	4
E2QS01	S1	5
E2QU01	Q1	6
E2QU02	Q2	7
E2QD02	D12/D21	8
E2QT02	T12	9
E2QS02	S2	10
E2QT01	T11/T13	11
E2QD01	D11/D22	12
—	—	13
—	—	14
—	—	15

Tabelle 3.1: Zuordnung physikalischer Geräte zu logischen Adressen.

ber besteht in dem Schutz der angeschlossenen Geräte vor Störspannungsspitzen auf den Übertragungsleitungen durch die galvanische Trennung von RS-232– und Leitungstreiber-Teil. Zerstörungen an den Geräten bei Stromausfällen sind nun sehr unwahrscheinlich. Die Daten werden sicher und störungsfrei übertragen. Bei dem 40°-System konnte diese Lösung übernommen werden, weitere Hardware-Modifikationen waren nicht erforderlich. Die Änderungen beschränkten sich auf die Anpassung der seriellen Gerätetreiber.

3.2.2 Integration der seriell angesteuerten Netzgeräte

Wie bereits ausgeführt, sind sowohl die Netzgeräte der alten Strahlführung als auch des 40°-Systems über serielle Leitungen ansteuerbar. Das heißt, Kommandos (im ASCII-Format) werden über die serielle Leitung an den Einplatinenrechner bzw. das 40°-System geschickt. Deshalb kann die Ansteuerung auch von einem Terminal vorgenommen werden.
Die Ansteuerung von Netzgeräten über eine serielle Leitung wurde jedoch nicht von der für die Ansteuerung der Netzgeräte zuständigen Software auf RECYC unterstützt. Damit konnten sie auch nicht von dem Bedienungsprogramm auf dem Rechner im Kontrollraum angesteuert werden. Es mußten daher zwei neue Gerätetypen mit entsprechenden Gerätetreibern eingeführt werden.
Jeder dieser Gerätetreiber bedient eine serielle Schnittstelle des Beschleuniger-Steuerungsrechners RECYC. Eine Schnittstelle ist mit dem Netzgerät des 40°-Systems verbunden, die andere mit dem Einplatinenrechner, der die Netzgeräte der alten Strahlführung bedient. Beide Gerätetreiber sind sich sehr ähnlich, da sie beide eine serielle Schnittstelle bedienen; sie unterscheiden sich jedoch bezüglich der gesendeten Kommandos und Datenformate. Nach der Entwicklung der Gerätetreiber wurden Namen für die Netzgeräte festgelegt und in die dafür vorgesehenen Tabellen eingetragen. Somit sind diese Netzgeräte nun wie jedes andere Netzgerät vom Kontrollraum aus ansprechbar.

4 Anwendung der neuen Architektur zur Überwachung der Magnete

Zur vollständigen Überwachung der Magnete reicht das Rücklesen der DAC-Werte (erste Ausbaustufe) nicht aus. Der Ausfall eines Netzgeräts konnte nur sehr indirekt durch Auswirkungen auf den Strahl festgestellt werden. Daher war für eine zweite Ausbauphase vorgesehen, die Magnetströme direkt zu messen. Die erforderliche Hardware zur Messung dieser Ströme war bereits vorhanden, da dies in der Konzeption der Netzgeräte berücksichtigt worden war.
Hauptaufgabe war daher die Inbetriebnahme und Einbindung in die Steuerprogramme, mit dem Ziel, eine schnelle Fehlerdiagnose zu ermöglichen.

4.1 Aufbau der Hardware

Jedes der verwendeten Netzgeräte verfügt über einen Ausgang, an welchem eine dem Magnetstrom proportionale Shunt-Spannung anliegt. Zur Erfassung dieser Shunt-Spannungen wurde ein im Institut konstruierter, mit 128 differentiellen Eingängen ausgerüsteter 12-Bit Analog-Digital-Konverter (ADC) eingesetzt. Dieser ADC ist über ein Q-Bus-Interface mit RECYC verbunden. Die Register des ADCs sind dort in einen Speicherbereich abgebildet und können gelesen und beschrieben werden.
Bei der ersten Inbetriebnahme des ADCs konnten keine reproduzierbaren Ergebnisse erzielt werden, sodaß zunächst ein defekter ADC angenommen wurde. Genauere Untersuchungen ergaben jedoch, daß das Problem bei den Netzgeräten lag. Zur Reduzierung der Verlustleistung und des Kühlungsaufwands der Netzgeräte sind diese als Schaltnetzteile ausgeführt. Das bedeutet, die Ausgangsspannung wird nicht analog geregelt, sondern pendelt um einen eingestellten Mittelwert. Dies wirkt sich zwar wegen der hohen Schaltfrequenz (in diesem Fall ca. 300 kHz) und der Induktivität der Magnetspulen nicht auf die Konstanz der Magnetfelder aus, erschwert jedoch durch kapazitive und induktive Einstreuung des Leistungsausgangs auf den Shunt-Spannungs-Ausgang eine Messung. Eine Nachprüfung mit einem Oszilloskop ergab, daß dem Shunt-Spannungs-Ausgang eine Störspannungen in der Größenordnung der eingestellten Signalspannung (ca. 0.5 V_{SS}) überlagert ist.
Aus diesem Grunde wurde jeder der 128 differentiellen Eingänge des ADCs mit einem Tiefpaßfilter 2. Ordnung ausgestattet, dessen Grenzfrequenz 1kHz beträgt. Eine niedrigere Grenzfrequenz konnte nicht gewählt werden, da die Platzverhältnisse sehr beengt waren und kleine Bauteile benutzt werden mußten. Dies beschränkte die nutzbaren Kapazitäten und Induktivitäten. Die Widerstände konnten nicht zu hochohmig gewählt werden (100 kΩ), da sonst der hohe Eingangswiderstand des ADCs die Meßergebnisse verfälscht hätte.

Mit den angegebenen Werten beträgt der maximale Meßfehler $\pm$ 2 Bit. Dies reicht völlig aus, da die primäre Aufgabe des ADCs die Funktionskontrolle der Netzgeräte und DACs ist. Eine Einstellung der Magnete über die Shunt-Spannungen war nie vorgesehen. Dies hätte wesentlich genauere ADCs erfordert, denn schon geringe Veränderungen der Einstellung bei einigen Magneten in der Größenordung 2^{-14} bis 2^{-13} des Maximalwertes beeinflussen empfindlich die Strahllage.

4.2 Integration in die Softwarearchitektur

Zur Speicherung der Parameter jedes Geräts existiert auf RECYC eine globales Datensegment, der COMMON-Block, der eine verkettete Liste von Unit-Control-Blocks (UCBs) [3] enthält. Hier werden für jedes Gerät der eingestellte DAC-Wert, untere und obere Grenze für den DAC-Wert und andere wichtige Parameter abgelegt. Diese Datenstruktur ist zentral über ein Assembler-Macro definiert und daher leicht und transparent erweiterbar. Um die Softwarearchitektur des RECYC einheitlich zu halten, wurde in dem für die UCB-Struktur zuständigen Macro ein neuer Parameter (UN.PR7) definiert, in den die ADC-Werte geschrieben werden. Da die UCB-Datenstruktur in das globale Datensegment, den COMMON-Block abgebildet ist, mußte auch das LCP, das den COMMON-Block auf angeschlossene Workstations überträgt, modifiziert werden. Gleiches gilt für die Definition des COMMON-Blocks auf den Workstations.

Nicht jedes Gerät muß notwendigerweise mit einem ADC-Kanal verbunden sein. Um zu erkennen, welche Geräte gültige ADC-Werte gelesen haben wurde im Flag-Wort (UN.FLG) eines UCBs ein neues Flag eingeführt (UN.ADC). Ist dieses Flag gesetzt, sind die ADC-Werte gültig, im anderen Fall nicht. Die Zuordnung zwischen Gerät und ADC-Kanal, die über steckbare Kabelverbindungen erfolgt, ist in einer Datei (ADCZUORD.DAT im Verzeichnis DL1:[1,1]) dokumentiert. Diese ASCII-Datei kann bei einer Veränderung leicht durch Editieren modifiziert werden. Dabei erfolgt die Zuordnung zwischen Gerät und ADC-Kanal über den Gerätenamen. Während der Initialisierung der Steuerprogramme wird diese Datei eingelesen und in eine systemnahe Form umgewandelt.

Um ständig über den aktuellen Stand der ADC-Werte zu verfügen, werden diese bei zwei verschiedenen Ereignissen gelesen. Zum einen bei der kompletten Versendung des COMMON-Blocks, die in Abständen von 50 Sekunden erfolgt und zum anderen bei jeder Versendung eines Netzgeräte-Parameter-Blocks (UPDATE-Block). Damit ist gewährleistet, daß der Ausfall eines Magneten spätestens nach 50 Sekunden erkannt werden kann. Die Erkennung eines Fehlers erfolgt halbautomatisch mithilfe des Programms ADC_MCP, das im Anhang näher beschrieben wird. Auf eine vollautomatische Fehleranzeige wurde verzichtet, da die Auflösung des ADCs im Bereich kleiner eingestellter DAC-Werte für eine sichere Fehlererkennung nicht ausreicht. Der Operateur würde ständig Fehlermeldungen erhalten,

die keinen realen Fehlersituationen entsprechen würden.

Die ADC-Werte sollten in den Control-Boxen[5] des MCPs angezeigt werden; damit wären alle Parameter eines Magneten auf einen Blick überschaubar gewesen. Dies hätte jedoch größere Modifikationen an undokumentierten internen Funktionen des MCPs bedeutet. In der nächsten Version des MCPs, die zur Zeit in Planung ist, wird die Anzeige der ADC-Werte jedoch integriert sein.
Als Zwischenlösung dient das Programm ADC_MCP. Hierbei wird die Control-Box-Anzeige des MCPs gespiegelt, in den Control-Boxen werden jedoch nur die ADC-Werte angezeigt. Wenn ein Operateur einer Control-Box der MCP-Benutzeroberfläche ein anderes Gerät zuordnet, wird diese Zuordnung auch in ADC_MCP durchgeführt, sodaß die angezeigten ADC-Werte immer der Zuordnung des MCP entsprechen. Dieses Programm benutzt die Screen-Managment-Funktionen des Betriebssystems zur fenster-orientierten Anzeige der Werte.

[5] Control-Boxen sind graphische Elemente des MCPs, die der Anzeige der Parameter eines Netzgeräts dienen.

5 Ergebnisse und Ausblick

Durch die Integration des 40°-Systems in die Beschleunigersteuerung ist es nun möglich, dieses vom Kontrollraum aus zu bedienen. Während eines mehrwöchigen Strahlbetriebs konnte die einwandfreie Funktion dieses Systems verifiziert werden.

Nach der Fertigstellung der für die Client-Server-Architektur notwendigen Komponenten war es möglich, die ersten Erfahrungen mit dieser neuen Kommunikations-Schnittstelle zu sammeln, wobei sie ihre volle Funktionsfähigkeit und praxisgerechte Eignung unter Beweis stellte. Die Datenübertragung erfolgte hierbei schnell und sicher und belastete den Rechner nur wenig. Die Client-Funktionsbibliothek erwies sich bei der Entwicklung des Programms zur Überwachung der Magnet-Ströme als äußerst nützlich und beschleunigte dessen Fertigstellung um mehrere Wochen. Das Programm zur Überwachung der Magnet-Ströme, das die neue Schnittstelle benutzt, konnte während eines Tests erfolgreich zur Ermittlung defekter Netzgeräte eingesetzt werden und stellte ein große Erleichterung dieser Arbeit dar.

Für die Zukunft ist eine Umgestaltung des Kontrollraums in Vorbereitung. Hierbei sollen die vorhandenen Systeme und Anzeigen durch einen leistungsfähigen zentralen Rechner mit zwei angeschlossen X-Terminals ersetzt werden. Dadurch wird es möglich, alle, die Beschleunigersteuerung und –überwachung betreffenden Programme mit einer graphischen Benutzeroberfläche auszustatten, die eine einfache und übersichtliche Bedienung ermöglicht. Die erstellte Client-Server-Architektur ist ein wichtiger Bestandteil des neuen Kontrollraum-Konzepts, da sie es ermöglicht, alle für die Beschleunigersteuerung und –überwachung zuständigen Programme auf einem Rechner zu betreiben.

Auch im Hinblick auf eine mögliche Ersetzung des alten Beschleuniger-Steuerungsrechners (RECYC), einer PDP 11/73, durch eine VAX unter VMS oder VAXELN und des damit notwendigen Neuentwurfs der Beschleunigersteuerung wäre die Client-Server-Architektur von Vorteil: Programme, welche die Server-Schnittstelle benutzen, müssen bei einem Wechsel des RECYC nicht umgeschrieben werden, lediglich der Server muß an die neuen Protokolle angepaßt werden. Der Server unterstützt bereits jetzt alle Funktionalitäten, die eine Modernisierung der Beschleunigersteuerung beinhalten sollte.

In Vorbereitung befindet sich zur Zeit ein Testprogramm zur weiteren Vereinfachung des Netzgerätetests, das ebenfalls die Server-Schnittstelle benutzen wird. Mit diesem Programm sollen die Kennlinien aller an den ADC angeschlossenen Netzgeräte gemessen und archiviert werden können. Die gemessenen Daten werden in einer für PLOTDATA (ein Programm zur Meßdatenauswertung) lesbaren

Form in Dateien abgelegt und lassen sich graphisch darstellen, sodaß sich ein schneller Überblick über die Funktionsfähigkeit der Netzgeräte ergibt.
Sobald die beiden X-Terminals im Kontrollraum installiert sind, ließe sich eine Schnelltest-Funktion implementieren, welche die Kennlinie eines bestimmten Netzgeräts aufnimmt und sofort graphisch darstellt. Damit wären schnelle Kontrollen durchführbar.

A Client-Server-Kommunikation

Im folgenden Kapitel wird näher auf die zur Client-Server-Kommunikation gehörenden Datenstrukturen, Kommandos und Datentypen eingegangen. Ferner werden die vorgefertigten Kommunikations-Prozeduren der Client-Funktionsbibliothek vorgestellt.

A.1 Aufbau eines Datenpaketes

Um eine größtmögliche Flexibilität im Datenaustausch zu erreichen, werden die Informationen als ASCII-kodierte Zeichenkette übergeben. Kommandos und Daten sind direkt lesbar. Zur Auswertung sind daher die standard-'C' Ein/Ausgabe-Funktionen *sprintf* und *sscanf* [17], die auf Zeichenketten angewendet werden, benutzbar. Damit ist eine formatfreie Datenübertragung möglich, sodaß ein breites Spektrum an Daten– und Gerätetypen abgeckt werden kann.
Eine Zeichenkette ist immer gleich aufgebaut: Sie besteht aus einem oder mehreren Wörtern. Wort bedeutet in diesem Zusammenhang eine durch Leerzeichen, Zeilenanfang oder Zeilenende abgegrenzte Zeichenkette. Im ersten Wort des Datenpaketes steht das Kommando, optional gefolgt von einem oder mehreren Geräteblöcken oder einem Text:

Kommando [Geräteblock [...] | Text]

Ein Geräteblock besteht aus einem Gerätenamen, gefolgt von einer durch diesen Namen festgelegten Anzahl von Zahlen oder einer Zeichenkette, die die Datentypen eines Gerätes festlegt.

Geräte-Name [Zahl [...] | Datentypen]

Die Datentypen-Zeichenkette bestimmt Art und Anzahl der zu einem Gerät gehörenden Daten. Jedes Zeichen dieser Zeichenkette steht für ein Datenelement.

Die Bedeutung dieser Zeichen ist folgende:

'I' Long-Integer, Read Only

'J' Long-Integer, Setable

'F' Floating-Point, Read Only

'G' Floating-Point, Setable

Drei Beispiele sollen das oben Erklärte verdeutlichen:

```
DNAM E1BM01 JIJ E1BM02 JIJ S1QU05 JIJI
```

Das erste Beispiel stellt die Übertragung eines Teils der Gerätenamen und Datentypen dar. Es werden drei Gerätenamen und ihre Datentypen definiert. Da es sich bei den beiden ersten Geräten um Standard-Dipolmagneten handelt, sind die Datentypen gleich. *JIJ* bedeutet, zu dem Gerät gehören drei Daten. Das erste Datum (J) ist der DAC-Wert des Netzgerätes, dieser ist Setable, kann also verändert werden. Das nächste Datum ist der zugehörige ADC-Wert, er kann nur gelesen aber nicht verändert werden. Das letzte Datum stellt den Gerätezustand dar, in ihm stehen Informationen darüber, ob das Netzgerät angeschaltet ist und dergleichen. Dieser Wert ist ebenfalls Setable[6].
Bei dem letzten Gerät handelt es sich um einen Quadrupolmagneten. Hier wird ein weiterer Parameter benötigt, welcher die Polung der Magnetwicklung angibt. Dieser Parameter kann positiv, negativ oder null sein, je nachdem, ob der Feldgradient ein positives oder negatives Vorzeichen hat oder nicht bekannt ist. Wichtig ist dieser Parameter bei der Berechnung des Feldgradienten aus den eingestellten DAC-Werten für das Strahllagen-Simulationsprogramm XBEAM.

```
DDAT E1BM01 1234 3050 8192 E1BM02 4321 5030 8193
```

Bei diesem Beispiel handelt es sich um eine Datenübertragung vom Server zu einem Client. Die Reihenfolge der Daten entspricht der im ersten Beispiel.

```
SDEV E1BM01 1234 8192 E1BM02 4321 0
```

Das letzte Beispiel zeigt die Benutzung des Multi-Set-Kommandos. Hier werden zwei Magnete auf verschiedene Werte gesetzt. Zu jedem dieser Magnete gehören zwei Daten, da nur zwei setable sind (gemäß Beispiel 1).

[6]Nur das Bit für An/Aus in diesem Flag-Wort ist setzbar

Bisher sind zwei Gerätetypen implementiert:

Standard-Magnet `'JIJ' --- DAC-Wert, ADC-Wert, Flags`

Quadrupolmagnet `'JIJI' --- DAC-Wert, ADC-Wert, Flags, Polung`

Der zusätzliche Parameter 'Polung' bei Quadrupolmagneten wurde eingeführt, weil manche Quadrupole in ihren elektrischen Anschlüssen vertauscht sind. Bei der Berechnung des Feldgradienten ist die Polung jedoch wichtig, da damit das Vorzeichen bestimmt wird. Der Parameter kann drei verschiedene Werte haben:

'0' `Polung unbekannt.`

'1' `Polung unvertauscht, positives Vorzeichen des Feldgradienten.`

'-1' `Polung vertauscht, negatives Vorzeichen des Feldgradienten.`

A.2 Geräte-Datenstruktur

Die Datenstruktur DEVICE, in der Geräte-Beschreibung und Daten abgelegt sind, wird in *MBX_DEF.H* definiert und in den Funktionsbibliotheken *CLIENT-_SKELETT.C* und *CLIENT_AST.C* benutzt. Sie ist dynamisch angelegt, das heißt nicht auf eine bestimmte Anzahl von Geräten festgelegt.

```
#define DEVMAXDATA 14

typedef struct tagDEVICE *DEVICEptr;
typedef struct tagDEVICE DEVICE;

struct tagDEVICE {
    char        name[9];        /* 8 Zeichen + '\0' am Ende     */
    short       max_data;       /* Anzahl der Daten bei diesem
                                Geraet                           */
    char        updated;        /* Flag fuer Updated            */
    char        typ[DEVMAXDATA];/* Typ der einzelnen Daten:
                                   'J': long, setable
                                   'I': long, read only
                                   'F': float, read only
                                   'G': float, setable
                                */
    union {
        float   f_value;        /* float-wert                   */
        long    l_value;        /* oder long-wert               */
        }       data [DEVMAXDATA];/* und das DEVMAXDATA-mal     */
    DEVICEptr   next;           /* link zum naechsten element der
                                        liste                    */
    };
```

Das Character-Array *name* enthält den Namen des Gerätes. Dieser darf maximal 8 Zeichen lang sein, das neunte Zeichen ist gemäß 'C'-String-Konvention auf 0 zu setzen.
Das Wort *max_data* gibt an, wieviele Daten zu diesem Gerät gehören. Für standard Magnet-Netzteile wären dies 3 Daten, für die HF-Regelung jedoch 12 Werte. Bis zu *DEVMAXDATA* Daten sind hier pro Gerät möglich. Diese Konstante steht zur Zeit auf 14, sollten mehr Daten pro Gerät benötigt werden, läßt sie sich anpassen.
Das Feld *updated* ist ein Flag. Es zeigt an, ob im *data*-Feld aktuelle Daten abgelegt wurden. In der *write_device*-Funktion wird dieses Flag benutzt, um zu bestimmen, welche Gerätedaten gesendet werden sollen.

Die Felder *typ* und *data* gehören zusammen. In *typ* wird festgelegt, welchen Datentyp *data* hat. Die Möglichkeiten sind hier Long-Integer und Floating-Point, beide setzbar (setable) oder 'nur lesbar' (read only).
Den Abschluß bildet *next*. Hier handelt es sich um die Verbindung zum nächsten Gerät.

A.3 Client-Funktionsbibliothek

Die Funktionsbibliothek besteht aus zwei Programmodulen:

CLIENT_SKELETT.C Enthält die Kommunikations-Funktionen.

CLIENT_AST.C Enthält ein Skelett der Kommunikations-AST-Routine.

Beide Programmodule befinden sich im Verzeichnis [MCP$ROOT.ADCUSER]. *CLIENT_SKELETT.C* muß mit #include eingebunden werden. *CLIENT_AST.C* kopiert man sich am besten in sein Programm, denn hier muß der Programmierer die Schnittstelle an seine Bedürfnisse anpassen.
Um dem Benutzer die Arbeit zu erleichtern, funktioniert die Datenübertragung zum und vom Server vollautomatisch mit den Prozeduren

```
write_device (DEVICEptr root_device);
read_device (DEVICEptr root_device, char* string);
build_device_list (DEVICEptr root_device, char* string);
```

Erstere Prozedur erwartet eine verkettete Liste von DEVICE-Elementen, deren Wurzel *root_device* ist.
Alle Elemente, deren *updated*-Flag gesetzt ist, werden zum Server übertragen, wobei zu beachten ist, daß nur die benötigten Daten übertragen werden. So macht es keinen Sinn, nicht setzbare Daten zu übertragen, es werden nur die setzbaren übertragen. Die benutzte verkettete Liste kann die Gleiche sein, die auch die vom Server übertragenen Daten enthält, sie muß aber nicht. Das bietet den Vorteil, daß eine Liste mit zu setzenden Geräten vorbereitet werden kann, ohne daß es zu Störungen durch Updates vom Server kommt. Diese Liste kann ein Spiegelbild der Empfangsliste oder gänzlich anders aufgebaut sein, wichtig ist hierbei nur, daß sie aus DEVICE-Elementen gebildet wird. Mit dieser Funktion können einzelne, mehrere oder auch alle Geräte auf einmal gesetzt werden.
Die *read_device*-Prozedur wertet einen String aus, der das Format eines Daten-Strings aus einer Client-Mailbox hat. Die zu den Geräte-Namen gehörenden Daten werden automatisch in der Geräte-Liste abgelegt, deren Anfang durch *root_device* gegeben ist. Welche Daten erwartet werden, bestimmt das Feld *typ* in DEVICE; dort wird eine ASCII-Zeichenkette abgelegt, deren Elemente die Typen für das *data*-Feld angeben.

Die Prozedur *build_device_list* wertet den Messagetyp MBX_DEVICE_NAMES aus. Bei der Auswertung können zwei Fälle eintreten: Zum Einen kann der Gerätename ein bis dato unbekanntes Gerät bestimmen. In diesem Fall wird der Geräteliste ein weiteres Element angefügt. Sollte der Gerätename jedoch bereits vorhanden sein, wird die Datentyp-Definition mit der neuen Definition überschrieben. Dieses Überschreiben der alten Definitionen läßt eine Änderung während des laufenden Betriebs zu. Geräteparameter können so hinzugefügt werden, ohne Programme, welche diese Parameter nicht benutzen, abbrechen und neu starten oder gar compilieren und linken zu müssen.
Die *write_device*-Prozedur versucht, so viele Geräte pro QIO zu übertragen, wie in einen Kommando-Block passen, um möglichst wenige QIOs zu benutzen. Das spart viel Zeit, da der Hauptzeitfaktor bei der Mailboxkommunikation die Ausführung des QIOs ist und nicht so sehr die Größe des übertragenen Datenblocks. Der Server benutzt für die Datenübertragung zu den Clients die gleiche Methode. Weitere Funktionen sind:

```
mailbox_open ();
mailbox_init ();
mailbox_send_command (MBX_DATA_REQUEST buffer);
convert_dac_to_field (DEVICEptr root_device);
convert_field_to_dac (DEVICEptr root_device);
```

Die Prozedur *mailbox_open ()* assigned die Kommando-Mailbox[7], erzeugt die Client-Datenmailbox CLIENT$⟨pid⟩ (wobei ⟨pid⟩ die hexadezimal ausgegebene achtstellige Process IDentifikation Number[8] ist), eröffnet die Kommunikation zum Server, wartet auf die Rückmeldung und fordert die Gerätenamen und Datentyp-Definitionen an.
Die Prozedur *mailbox_init ()* aktiviert den Hello-Timer[9] und den Lese-QIO auf die CLIENT$⟨pid⟩-Mailbox.
Mit *mailbox_send_command (...)* können Kommandos an den Server geschickt werden. Welche Kommandos das sind, beschreibt das folgende Kapitel.
Die Prozedur *convert_dac_to_field (DEVICEptr root_device)* konvertiert DAC-Werte in Magnetfeldstärken, die in Parameter *data[14].f_value* abgelegt werden. Zur Umrechnung wird die Datei *feld_eichung.dat* im Verzeichnis *[MCP$ROOT.ADCUSER]* benutzt. In jeder Zeile dieser Datei stehen Gerätename, multiplikative und additive Konstante gefolgt von einem optionalen Kommentar.

[7] Die Kommando-Mailbox trägt den Namen ADCMAILIN und wird vom Server gruppenweit erzeugt.
[8] Die PID in dem Mailboxnamen bietet den Vorteil, mit dem DCL-Kommando SHOW LOGICAL/GROUP sofort sehen zu können, welche Mailbox zu welchem Programm gehört. Zudem ist sie eine eindeutige Zahl.
[9] Der Hello-Timer schickt in periodischen Abständen Aktivitäts-Meldungen an den Server

Die Prozedur *convert_field_to_dac (DEVICEptr root_device)* führt die umgekehrte Konvertierung durch.
Diese beiden Prozeduren werden gebraucht, wenn ein Programm statt Magnetfeldern DAC-Werte benötigt und benutzt. XBEAM ist hierfür ein Beispiel, da für eine Simulation der Strahllage die Magnetfelder wichtig sind.

A.4 Kommandotypen

Es folgen nun die Kommando-Typen, die Clients an den Server senden können.

MBX_GET_DEVICES_NAMES

Veranlaßt den Server, die Geräte-Namen und Datentyp-Definitionen zu senden.

Code: "GNAM"

Parameter: Keine

Antwort-Typ: Mehrere MBX_DEVICE_NAMES-Messages. Zum Schluß eine MBX_LAST_NAME-Message, um anzuzeigen, daß die Geräteliste jetzt vollständig ist.

MBX_GET_COMMON_BLOCKS

Trägt den Client in die *new_common*-Liste ein. Neu eingetroffene COMMON-Blöcke werden an den Client geschickt.

Code: "GCBL"

Parameter: Keine

Antwort-Typ: Mehrere MBX_DATA_BLOCK-Messages. Nach Senden aller Gerätedaten folgt als Abschluß eine MBX_LAST_COMMON-Message. Bei jedem Geräteeintrag, für den Daten empfangen wurden, wird das updated-Flag gesetzt.

MBX_GET_UPDATES

Der Client wird in die *update*-Liste eingetragen. Im Server eingetroffene UPDATE-Blöcke werden an den Client weitergereicht.

Code: "GUPD"

Parameter: Keine

Antwort-Typ: Ein oder mehrere MBX_DATA_BLOCK-Messages. Das Format ist das Gleiche wie bei der Versendung eines COMMON-Blocks. Bei jedem Geräteeintrag, für den Daten empfangen wurden, wird das updated-Flag gesetzt.

MBX_GET_CONTROL_BLOCKS

Veranlaßt den Server, die Zuordnung Control-Box ↔ Gerätename an den Client zu schicken und diesen in die *control*-Liste einzutragen. Bei jeder Änderung dier Zuordnung dieser Client Benachrichtigt wird.

Code: "GCTL"

Parameter: Keine

Antwort-Typ: Ein oder mehrere MBX_CONTROL_BLOCK-Messages. Das Format einer solchen Message ist *DCTL Gerätename Control-Nummer [...]*.

MBX_SET_CONTROL

Verändert die Zuordnung Control-Box ↔ Gerätename.

Code: "SCTL"

Parameter: Gerätename, Control-Nummer

Antwort-Typ: Alle in der *control*-Liste eingetragenen Clients erhalten MBX_CONTROL_BLOCK-Messages.

MBX_GET_SINGLE_CHANNEL

Im Gegensatz zu MBX_GET_UPDATES wird der Client hier nur über Änderungen in einem bestimmten Gerät unterrichtet. Dies dient der Begrenzung der Kommunikationsbandbreite bei Clients, die nur an bestimmten Geräten Interressiert sind. Der Client wird in die *single_channel*-Liste eingetragen.

Code: "G1CH"

Parameter: Gerätename

Antwort-Typ: MBX_DATA_BLOCK, wie bei MBX_GET_UPDATES.

MBX_UNGET_SINGLE_CHANNEL

Macht einen durch MBX_GET_SINGLE_CHANNEL getätigten Eintrag in der *single_channel*-Liste rückgängig.

Code: "U1CH"

Parameter: Gerätename

Antwort-Typ: Keiner.

MBX_SET_DEVICE

Veranlaßt den Server, ein oder mehrere Gerät(e) auf einen oder mehrere bestimmte(n) Wert(e) zu setzen. Multiset-Kommando.

Code: "SDEV"

Parameter: Gerätename Daten [...]

Antwort-Typ: Keine explizite Antwort.

MBX_OPEN_COMMUNICATION

Teilt dem Server mit, daß ein Client hinzu gekommen ist.

Code: "OPEN"

Parameter: Keine

Antwort-Typ: MBX_ACKNOWLEDGE. Bestätigung des Servers, daß der Client bekannt und die Mailbox zu ihm geöffnet ist.

MBX_CLOSE_COMMUNICATION

Beendet die Kommunikation mit dem Server. Der Client wird aus allen Listen ausgetragen, seine Mailbox wird geschlossen.

Code: "CLOS"

Parameter: Keine

Antwort-Typ: Keine Antwort.

MBX_SAY_HELLO

Teilt dem Server mit, daß ein Client noch aktiv ist.

Code: "HELO"

Parameter: Keine

Antwort-Typ: Keine Antwort.

MBX_REQUEST_ONE_COMMON

Wie MBX_GET_COMMON_BLOCKS, jedoch wird hiermit ein COMMON-Block von RECYC angefordert.

Code: "RCBL"

Parameter: Keine

Antwort-Typ: Mehrere MBX_DATA_BLOCK-Messages und am Ende eine MBX_LAST_COMMON-Message.

MBX_GET_ERROR_MESSAGES

Trägt den Client in die *error_messages*-Liste ein. Sollten Fehlermeldungen des RECYC im Server ankommen, werden sie an diesen Client weitergereicht.

Code: "GERR"

Parameter: Keine

Antwort-Typ: Eine MBX_ERROR-Message.

A.5 Message-Typen

Es folgen nun die Message-Typen, die der Server an Clients schicken kann.

MBX_CONTROL_BLOCK

Zuordnung Control-Box $\leftrightarrow$ Gerätename.

Aufbau: *DCTL Gerätename Control-Nummer [...]*

MBX_DATA_BLOCK

Gerätedaten.

Aufbau: *DDAT Geräteblock [...]*

MBX_DEVICE_NAMES

Gerätenamen und Datentyp-Definitionen.

Aufbau: *DNAM Gerätename Datentypen [...]*

MBX_LAST_NAMES

Letzte Geräte-Definition empfangen.

Aufbau: *DLNA*

MBX_LAST_COMMON

Letzten Datenblock einer kompletten Datenübertragung empfangen.

Aufbau: *DLCO*

MBX_ERROR

Fehler-String

Aufbau: *DERR text*

MBX_ACKNOWLEDGE

Server hat diesen Client akzeptiert und die Client-Datenmailbox geöffnet.

Aufbau: *DACK*

B ADC-Überwachungsprogramm

Zur Überwachung der Shunt-Spannungen dient das Programm ADC_MCP. Es bildet die Control-Box-Struktur des MCP nach und ist fensterorientiert und menügesteuert.
In acht Control-Boxen, analog zu der Anzeige der DAC-Werte in den Control-Boxen des MCP, werden die zugehörigen ADC-Werte der Shunt-Spannungen angezeigt. Hierdurch läßt sich recht einfach erkennen, ob die zugehörigen Netzgeräte defekt sind.
Um die Fehlererkennung noch komfortabler zu gestalten, ist eine zusätzliche Funktion eingebaut, mit welcher der Operateur die strahlführenden Elemente einer Beschleunigersektion testen kann. Unter Beschleunigersektionen wird hier die von [6] definierte Aufteilung der strahlführenden Elemente in logische Gruppen verstanden. Die Bezeichnung der Sektionen entspricht den ersten beiden Zeichen der Gerätenamen.
Zum testen der Shuntspannungen wird mit den Cursor-Tasten der Menüpunkt **CHECK** angewählt. In die erscheinende Dialogbox wird die Sektion eingetragen und derjenige Bereich der Shunt-Spannung, der für einen Defekt charakteristisch ist. Nimmt man an, daß ein defektes Netzgerät keinen Strom mehr abgibt, so ist seine Shunt-Spannung 0 Volt. Durch die Digitalisierung dieser Spannung entsteht jedoch ein Meßfehler in der Größenordnung $\pm$ 8 Einheiten.
Alle Geräte, denen gültige ADC-Werte im angegebenen Bereich zugeordnet sind, werden in einem Fenster mit ihren Parametern angezeigt. Wird ein defektes Netzgerät gefunden, läßt sich seine Position in den Schaltschränken über den Menüpunkt **RACK** ermitteln. Dort wird der Gerätename eingegeben und die Positionsbeschreibung ausgegeben. Diese Beschreibung wird der Datei ng.txt im Verzeichnis RECYC::DL1:[30,1] entnommen und ist damit immer auf dem aktuellen Stand.
Ein weiterer Menüpunkt, **PROZEDUR**, erlaubt es, Batch-Jobs zu starten, deren Ausgabe in Fenster umgeleitet wird. Zum Starten eines Batch-Jobs wird eine Datei mit der Extension *.ext* gelesen. Die Datei ist folgendermaßen aufgebaut:

```
[Fenster-Ueberschrift]
Zeile Spalte Breite Hoehe [close | noclose] [hide | nohide]
[kommentar]
[kommentar]
[DCL-Kommando]
```

Die Kommandos *close* und *noclose* geben an, ob das Anzeige-Fenster bei Ende des Batch-Jobs geschlossen wird.
Mit dem Kommando *hide* läßt sich ein Batch-Job auch ohne Anzeige ausführen.
DCL-Kommando kann jedes gültige Kommando sein.

Zur Verdeutlichung folgt nun ein Beispiel:

```
Directory Fenster
10 5 60 8 close nohide
Hier soll ein Directory im Fenster angezeigt werden,
und das in drei spalten
$dir /column=3
```

In der ersten Zeile steht der Titel des neuen Fensters. Darunter Zeile, Spalte, Breite und Höhe des Fensters, gefolgt von der Anweisung, das Fenster nach Beenden des Kommandos zu schließen (close). Das Kommando *nohide* gibt an, daß eine Ausgabe auf dem Bildschirm erfolgt.
Die beiden verbleibenden Menüpunkte **QUIT** und **WINDOW** dienen zum Verlassen des Programms und zur Manipulation der Fenster.

C Erweiterungsplatine

Eine Erweiterungplatine besteht aus fünf Latch-Gruppen aus jeweils drei Latches des Typs 74HCT374, die von einem 4 → 16 Dekoderbaustein des Typs 74HCT154 angesteuert werden. Über ein Steckfeld lassen sich die fünf Latch-Gruppen jeweils einem Ausgang des Dekoders zuordnen.
Zur Vereinfachung ist nur eine der fünf Latch-Gruppen dargestellt.

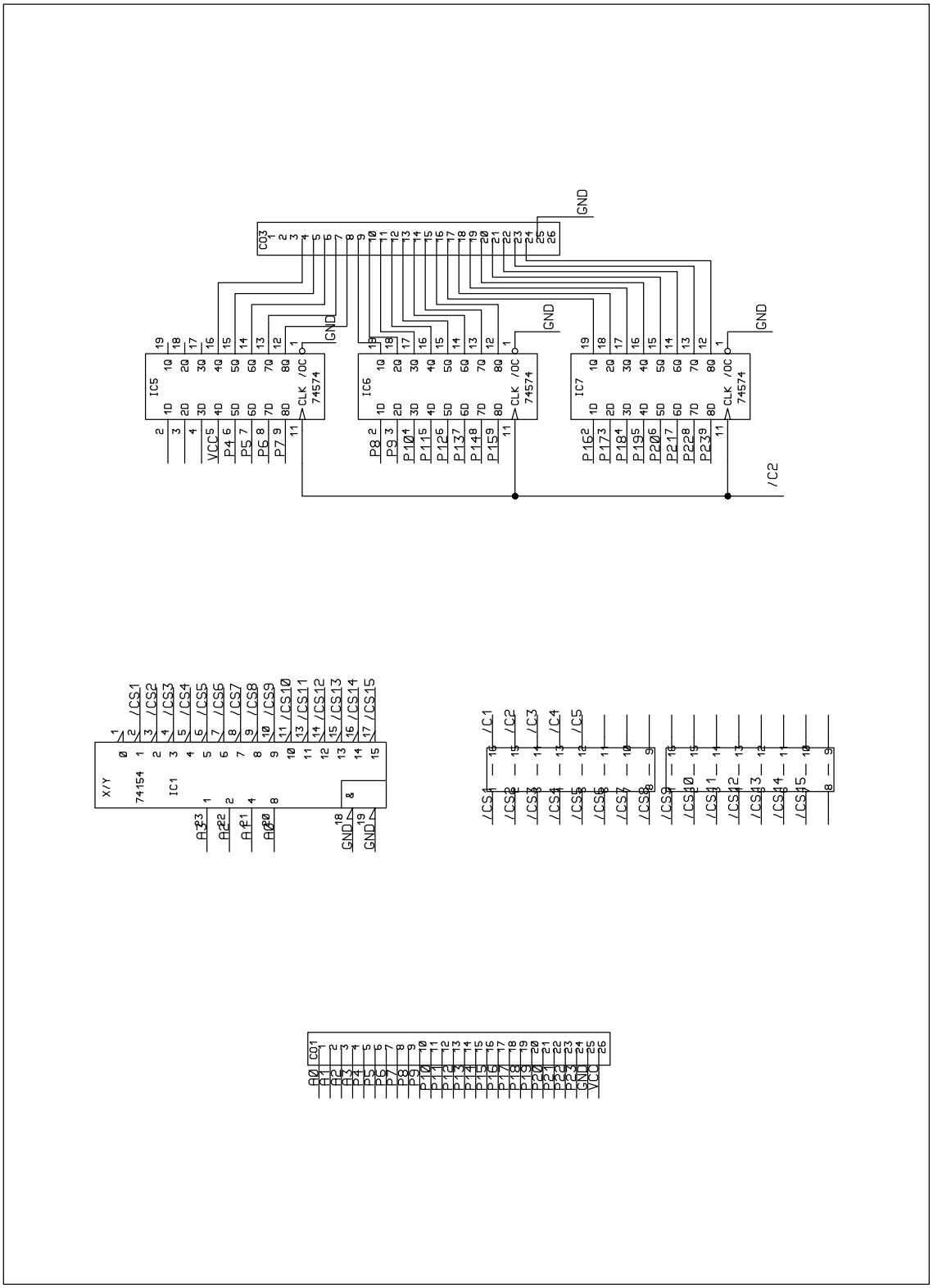

Abbildung C.1: Platine zur Erweiterung des Latch-Ausgangs des benutzten Einplatinenrechners. Über diese Platine können fünf Netzgeräte angesteuert werden.

Literatur

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

H.-D. Gräf, J. Horn, K.-D. Hummel, C. Lüttge, A. Richter, T. Rietdorf, K. Rühl, P. Schardt, E. Spamer, A. Stiller, F. Thomas, O. Titze, J. Töpper, H. Weise and T. Winkler, Proc. of the 1992 Linear Acc. Conf., Ottawa, Canada.

[2] H.-D. Gräf, H. Miska, O. Titze and Th. Walcher, Nucl. Instr. Meth. **153** (1978) 9;

Th. Walcher, R. Frey, H.-D. Gräf, E. Spamer and H. Theissen, Nucl. Instr. Meth. **153** (1987) 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, A. Schwierczinski, H. Theissen and O. Titze, Nucl. Instr. Meth. **153** (1978) 43.

[3] V.C. Huck, Dissertation D17, TH Darmstadt (1991), unveröffentlicht.

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

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

[6] V.C. Huck, Diplomarbeit, TH Darmstadt (1987), unveröffentlicht.

[7] R. Hamm, Diplomarbeit, TH Darmstadt (1991), unveröffentlicht.

[8] G. Kaster, Dissertation D17, TH Darmstadt (1989), unveröffentlicht.

[9] H. Weise, Diplomarbeit, TH Darmstadt (1988), unveröffentlicht.

[10] S. Richter, Diplomarbeit, in Vorbereitung.

[11] T. Winkler, Diplomarbeit, TH Darmstadt (1993), unveröffentlicht.

[12] Introduction to VMS System Services, Order Number: AA-CA69A-TE, (1988).

[13] VMS System Services Reference Manual, Order Number: AA-LA69A-TE, (1988).

[14] Guide to VMS Programming Resources, Order Number: AA-LA57A-TE, (1988).

[15] VMS IO User's Reference Manual: Part 1, Order Number: AA-LA84A-TE, (1988).

[16] K.D. Hummel, Diplomarbeit, TH Darmstadt (1987), unveröffentlicht.

[17] B. W. Kernigan & D. M. Ritchie, Programmieren in C.

[18] Maxim, New releases data book, (1990).

Danksagung

Zu guter Letzt möchte ich einigen Personen, die zu der Entstehung dieser Arbeit beigetragen haben, recht herzlich danken.

An erster Stelle danke ich Herrn Professor Dr. Achim Richter für die interessante und vielseitige Themenstellung. Die Möglichkeit in seiner Arbeitsgruppe am S–DALINAC mitzuarbeiten war mir ein besonderer Ansporn.

Für die bereitwillige Unterstützung bei allen Problemen mit den Rechnern danke ich Herrn Dr. O. Titze und Herrn Diplomphysiker J. Horn.

Desweiteren habe ich Herrn Dr. H. D. Gräf für seine ständige Diskussionsbereitschaft, insbesondere in allen Fragen der Strahlführung, zu danken.

Den Herren H. Diesener, M. Galemann, C. Lüttge, U. Nething, A. Stiller, T. Wesp und Frau S. Richter möchte ich für ihre Hilfe in allen Fragen der Physik und ihre ständige Gesprächsbereitschaft danken.

Bei den Mitarbeitern des Zeichenbüros möchte ich mich für ihre Hilfe bei der Fertigstellung dieser Arbeit bedanken.

Zum Schluß möchte ich allen bisher nicht genannten Mitarbeitern und Mitarbeiterinnen für das gute Arbeitsklima danken, daß entscheidend zum gelingen dieser Arbeit beigetragen hat.

Hiermit erkläre ich an Eides statt, daß ich die vorliegende Arbeit selbstständig verfaßt und nur die angegebenen Hilfsmittel eingesetzt habe.

Darmstadt, im Juli 1993

(Dirk Schmischke)