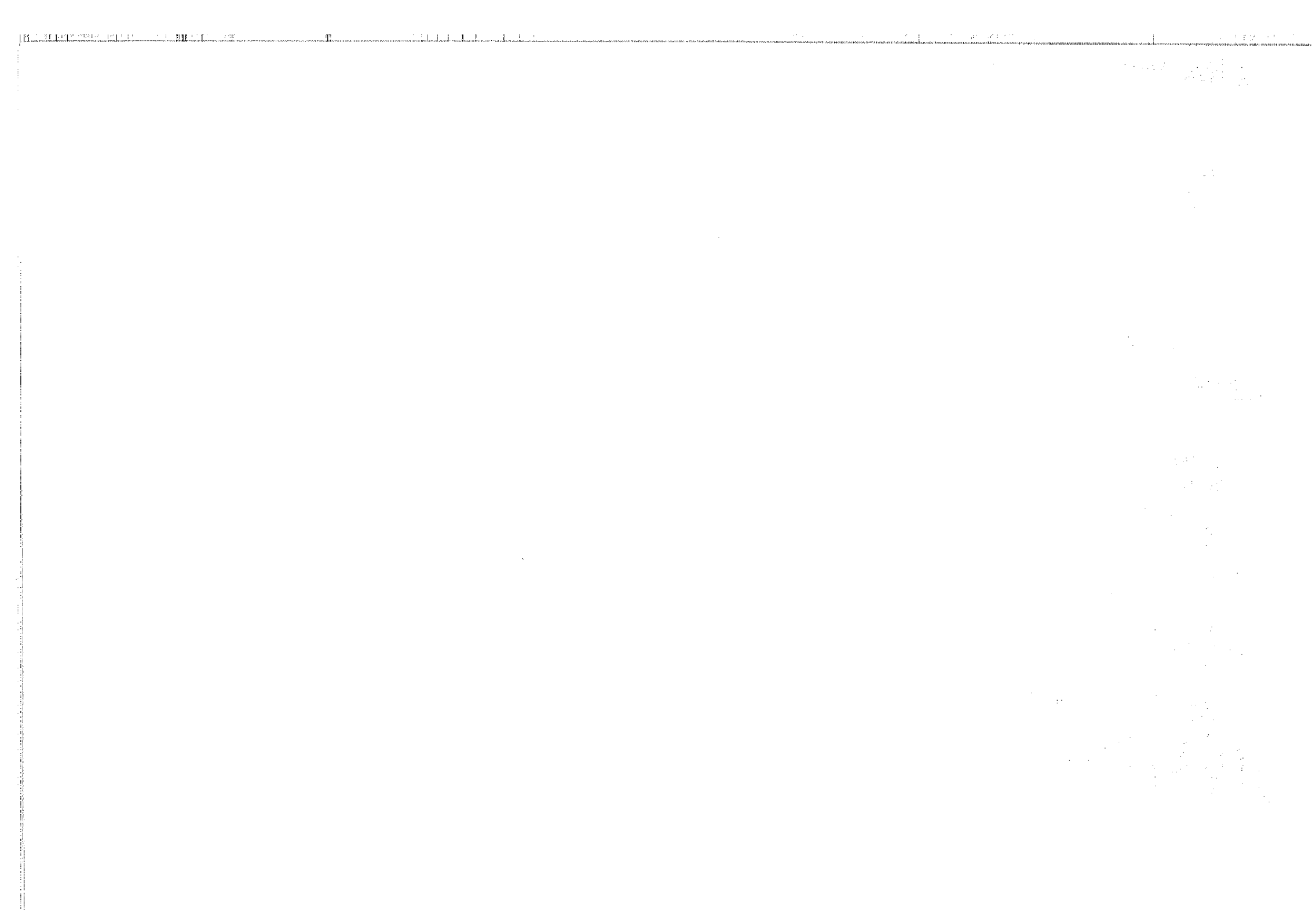
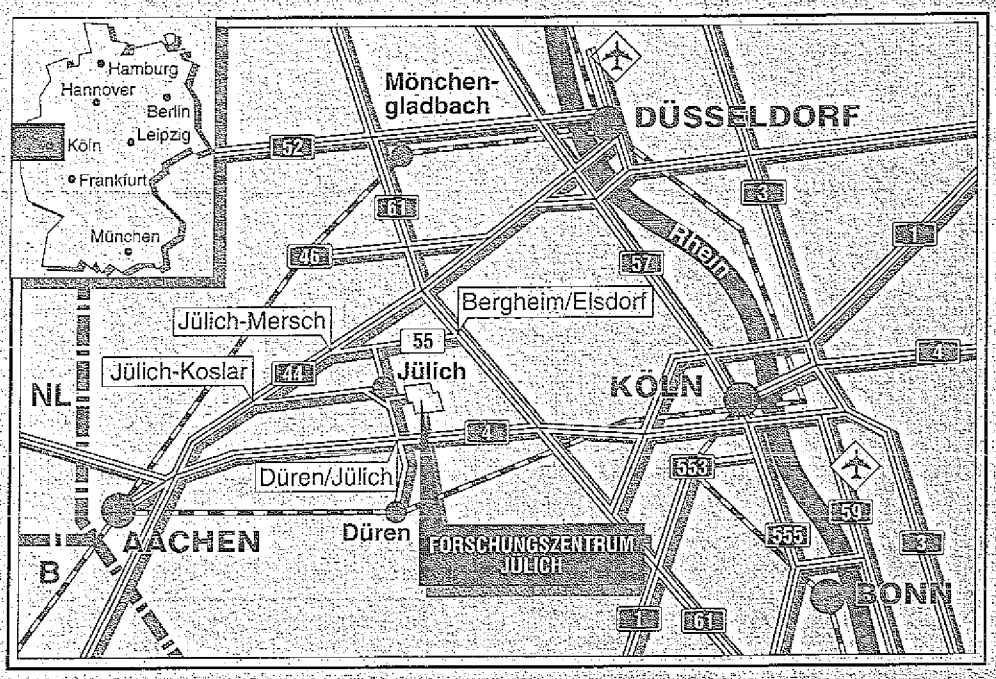
FORSCHUNGSZENTRUM JÜLICH GmbH

Zentrallabor für Elektronik

Strukturanalyse und Optimierung einer Vielbus Architektur zur Datenerfassung

Helmut Josef Kopp





Berichte des Forschungszentrums Jülich ; 3013
ISSN 0944-2952
Zentrallabor für Elektronik Jül-3013
D 82 (Diss. RWTH Aachen)

Zu beziehen durch: Forschungszentrum Jülich GmbH · Zentralbibliothek
D-52425 Jülich · Bundesrepublik Deutschland
Telefon: 02461/61-6102 · Telefax: 02461/61-6103 · Telex: 833556-70 kfa d

Strukturanalyse und Optimierung einer Vielbus Architektur zur Datenerfassung

Helmut Josef Kopp

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftliche Hilfskraft im Zentrallabor für Elektronik (ZEL) des Forschungszentrums Jülich in den Jahren 1992 bis 1994.

Mein Dank gilt meinem Betreuer Prof. Dr.-Ing. K. W. Pleßmann, Leiter des Lehr- und Forschungsgebiets für Verfahren der Prozeßdatenverarbeitung und Prozeßführung (pdv) an der Rheinisch-Westfälischen Technischen Hochschule Aachen. Ohne seine Anregungen und Hinweise wäre diese Arbeit nicht möglich gewesen. Herrn Prof. Dr.-Ing. W. Michaeli, Leiter des Instituts für Kunststoffverarbeitung an der Rheinisch-Westfälischen Technischen Hochschule Aachen, danke ich für das dieser Arbeit entgegengebrachte Interesse und die Übernahme des Korreferats.

Ich danke auch dem Institutsleiter des Zentrallabors für Elektronik, Herrn Dr. rer. nat. K. D. Müller, der mir durch seine Unterstützung die Möglichkeit gab, in seinem Institut als Doktorand zu arbeiten und der die Idee hatte, den PCI-Bus einzusetzen.

Mein Dank gilt auch Herrn Dr.-Ing. K. Zwoll, meinem Abteilungsleiter, der sich immer wieder für mich und meine Arbeit eingesetzt hat und mich besonders bei Fragen zur Systemarchitektur beraten hat.

Für die fachlichen Hinweise zur Physik danke ich Herrn Dr. rer. nat. M. Köhler und Herrn Dipl.-Phys. P. Wüstner.

Ebenso gilt mein Dank den zahlreichen Kollegen im Zentrallabor für Elektronik, die durch fachliche Hinweise und Mitarbeit geholfen haben.

Jülich, im September 1994 Helmut Kopp

Inhaltsverzeichnis

1. Einleitung 1
1.1. Atomphysik 1
1.2. Kernphysik 2
1.3. Elementarteilchenphysik 2
1.4. Das COSY-Jülich 3
1.4.1. Was kann mit COSY erforscht werden? 5
1.4.2. Datenerfassungssysteme am COSY-Jülich 6
1.5. Motivation 8
1.6. Übersicht 9
2. Strukturanalyse von DV-Systemen der Hochenergiephysik 10
2.1. Zusammenfassung 16
3. Die Physik des Experiments COSY 11 17
3.1. Zusammenfassung 21
4. Architektur des Datenerfassungssystems für COSY 11 22
4.1. Übersicht 22
4.2. Digitalisierungskomponenten 23
4.2.1. CAMAC-Überrahmen 23
4.2.2. FASTBUS-Überrahmen 23
4.3. Eventbuilder-Überrahmen 24
4.4. Datenflußstruktur des Datenerfassungssystems 25
4.5. Die Programmiersprache occam 2 26
4.6. Zusammenfassung 27
5. Formulierung des Datenerfassungsproblems in occam 2 28
5.1. Einführung 28
5.2. Allgemeine Darstellung der Prozesse zur Datenerfassung 29
5.2.1. Datenflußstruktur 29
5.2.2. Kontroll- und Datenfluß 30
5.3. Abbildung der Struktur in ein occam 2 Programm 34
5.3.1. Globale Deklarationen 34

5.3.2. Abbildung der EA{i}{j}-Prozesse in occam 2 35
5.3.3. Abbildung der VIC{i}-Prozesse in occam 2 35
5.3.4. Abbildung der SER{i}-Prozesse in occam 2 36
5.3.5. Abbildung des VME-Prozesses in occam 2 37
5.3.6. Abbildung des EB-Prozesses in occam 2 37
5.3.7. Abbildung des FORMAT-Prozesses in occam 2 37
5.3.8. Das Hauptprogramm 39
5.4. Die Prozesse des occam 2-Programms zur Datenerfassung 40
5.4.1. Die Event-Acquisitions-Prozesse 40
5.4.2. Der Subeventreader-Prozeß 43
5.4.3. Der Eventbuilder-Prozeß 48
5.5. Zusammenfassung 48
6. Implementierungskonzept für ein optimiertes System 49
6.1. Die optimierte Systemstruktur 49
6.2. Die optimierte Hardware 52
6.2.1. Auswahl des Prozessors 52
6.2.2. Auswahl des lokalen Busses 54
6.2.3. Zu entwickelnde Schnittstellen 55
6.3. Auswahl der Programmierumgebung für den C40 56
6.3.1. Beispiel für die Übersetzung: occam 2 zu parallel C 57
6.4. Beschreibung der optimierten Architektur 60
6.4.1. Die C40-Karte 62
6.4.2. Die VICbus-Karte 67
6.4.3. Die Aufsteckkarte 68
6.4.4. Das C40-Entwicklungssystem 68
6.4.5. Komplette Architektur 70
6.5. Zusammenfassung 74
7. Zusammenfassung 75
7.1. Erwartete Leistungsdaten 76
7.1.1. Maximale Datenerfassungsrate des VMEbus-Systems 76

7.1.2. Maximale Datenerfassungsrate des C40-Systems 77
8. Anhang **78**
8.1. Abkürzungen 78
8.2. Glossar 80
8.3. Das ZEBRA-Format 83
8.3.1. Typen der Logischen Records 83
8.3.2. Typen der Daten 83
8.3.3. Die ZEBRA-Dateistruktur 84
8.4. Die C40-Bus zu S_PCI-Bus Schnittstelle 85
8.4.1. Die C40-Bus zu S_PCI-Bus Master-Schnittstelle 85
8.4.2. Die C40-Bus zu S_PCI-Bus Slave-Schnittstelle 86
8.4.3. Die Adreßräume des C40- und des PCI-Busses 88
9. Literaturverzeichnis **103**

1. Einleitung

Zweck meiner Arbeit ist der Entwurf eines Datenerfassungssystems für Streuexperimente am Teilchenbeschleuniger und Speicherring COSY-Jülich (COoler SYnchrotron in Jülich). Bei einem Streuexperiment werden Teilchen zur Kollision gebracht. Dadurch können sie miteinander wechselwirken und es entstehen neue Partikel, wenn die Energie dazu ausreicht. Durch Analyse der Flugbahn und der Geschwindigkeit der Teilchen kann man Rückschlüsse auf deren innere Struktur ziehen. Der Nutzen eines Teilchenbeschleunigers läßt sich durchaus mit dem eines hochauflösenden Mikroskops vergleichen.
Um dem Leser eine Einführung in die Begriffe der Elementarteilchenphysik zu geben, grenze ich dieses Teilgebiet der Physik zunächst gegenüber der Atomphysik und der Kernphysik ab. Für eine ausführlichere Einführung sei auf /BETH 90/ /LANG 88/ und /MUSI 88/ verwiesen. Danach folgt eine kurze Beschreibung des COSY-Jülich. Den Abschluß dieser Einleitung bilden Abschnitte über die Motivation und eine Übersicht für die vorliegende Arbeit.

1.1. Atomphysik

Schon im Altertum machte sich der Mensch Gedanken über den Aufbau der Materie. Der Grieche Leukipp (etwa 480 - 420 v. Chr.) und sein Schüler Demokrit (etwa 460 - 370 v. Chr.), gelten als die geistigen Väter des Atomismus. Demokrit begründete durch philosophische Überlegungen die Existenz von unteilbaren Teilchen (griechisch: atomos = unteilbar).
Diese Theorie war aufgrund fehlender Beweise bis in die Neuzeit umstritten.
Es sollte das Hauptproblem der Physik werden, gefundene Modelle durch experimentelle Nachweise zu verifizieren.
Erst im 18.- und 19. Jh. wurde der Atomismus, z. B. von John Dalton (1766 - 1844), wieder aufgegriffen und bekam eine wissenschaftliche Grundlage. Chemiker postulierten, "... daß das Unterscheidungsmerkmal verschiedener Atome ihr Gewicht sei." /BETH 90/.
Eine Synthese der folgenden Entdeckungen verschiedener Chemiker und Physiker gelang erstmals Niels Bohr (1885 - 1962) durch sein Atommodell von 1913, für das er 1922 den Nobelpreis erhielt. Demnach besteht ein Atom aus einem Kern mit um ihn kreisenden Elektronen.
Durch erste Streuexperimente fand Rutherford 1914 heraus, daß die "Rückstoßatome", die beim Beschuß von Wasserstoff mit Elektronen gefunden wurden, Protonen sind. Durch ähnliche Versuche konnte später nachgewiesen werden, daß Atomkerne und auch Protonen eine innere Struktur besitzen.
Später stellte sich heraus, daß die Eigenschaften und die Erklärung der Hülle eines Atoms bzw. die Beschreibung des Kerns relativ stark voneinander entkoppelt

sind. Dadurch konnten sich die Atom- oder Hüllenphysik und die Kernphysik recht unabhängig voneinander entwickeln.

1.2. Kernphysik

"Die Kernphysik befaßt sich mit der experimentellen und theoretischen Erforschung der Eigenschaften und der Struktur der Atomkerne und aller kernphysikalischen Erscheinungen" /MUSI 88/. Der Atomkern kann in guter Näherung durch den Aufbau aus Nukleonen (Protonen, Neutronen) und durch Mesonen vermittelte Wechselwirkungen beschrieben werden. Nukleonen und Mesonen sind Elementarteilchen. Die Gesetze der Kernphysik sind daher durch die Elementarteilchenphysik begründbar.
Kern- und Elementarteilchenphysik sind also eng miteinander verknüpft.

1.3. Elementarteilchenphysik

"Die Elementarteilchenphysik, die oft auch Hochenergiephysik genannt wird, beschäftigt sich mit der experimentellen und theoretischen Erforschung der Elementarteilchen und ihrer Wechselwirkungen." /MUSI 88/.
Von 1930 bis 1964 führten Beobachtungen der kosmischen Höhenstrahlung, Experimente mit Nebelkammern und Beschleunigern zur Entdeckung immer neuer Teilchen. Dieser sogenannte "Teilchenzoo" wurde ursprünglich in Baryonen (schwere Teilchen), Mesonen (mittelschwere Teilchen) und Leptonen (leichte Teilchen) unterteilt.
Der entscheidende Durchbruch gelang 1964 den amerikanischen Physikern M. Gell-Mann und G. Zweig, die alle damals bekannten Mesonen und Baryonen auf drei Grundbausteine zurückführen konnten: Die Quarks. Baryonen werden aus Tripletts von Quarks, Mesonen aus Quark-Antiquark Dupletts gebildet. Aufgrund der starken Wechselwirkung zwischen den Quarks scheint es prinzipiell unmöglich zu sein, ein Quark zu isolieren.
Heute gelten Quarks und Leptonen (z. B. Elektronen und Neutrinos) als die eigentlich elementaren Teilchen. Die Wechselwirkungen werden durch die Feldquanten wie: Photonen, Gluonen und Vektorbosonen vermittelt.
Das Quark-Modell von 1964 mußte nach Experimenten an Teilchenbeschleunigern noch um drei Quarks erweitert werden.
Es gibt drei unterschiedliche Gruppen von Quarks, die sich u. a. in ihren Massen unterscheiden. Die erste Gruppe besteht aus dem "Up"- und dem "Down"-Quark. Up- und Down- Quarks sind die Bausteine der Materie unserer normalen Umwelt. Protonen und Neutronen (d. h. die Nukleonen) bestehen aus unterschiedlichen Tripletts von Up- und Down-Quarks. Die zweite Gruppe beinhaltet das "Strange"- und das "Charm"-Quark. Die dritte Gruppe bilden das "Bottom"- und "Top"-Quark. Die Existenz der Up-, Down-, Strange-, Charm- und Bottom-Quarks wurde durch Experimente an verschiedenen Teilchenbeschleunigern in der Zeit

von 1968 bis 1977 entdeckt oder bewiesen. Das Top-Quark wurde im April 1994 am amerikanischen Fermi National Accelerator Laboratory bei Chicago experimentell nachgewiesen /MOEL 94/.
Quarks der zweiten und dritten Gruppe kommen in Baryonen oder Mesonen in der Natur normalerweise nicht vor. Sie müssen - für kontrollierbare Experimente - durch Beschleuniger künstlich erzeugt werden.
Diese Erkenntnisse konnten erst durch den Einsatz von Teilchenbeschleuniger-anlagen im GeV (Giga Elektronenvolt) Bereich gewonnen werden. Dabei wurden Streuexperimente realisiert, die oft durch räumlich große und komplex aufgebaute Detektoren meßtechnisch erfaßt wurden.

1.4. Das COSY-Jülich

Am 1. April 1993 wurde das COSY-Jülich offiziell eingeweiht. COSY steht für COoler-SYnchrotron und ist ein Teilchenbeschleuniger und Speicherring mit Phasenraum-Kühlung für Protonen oder Ionen.
CO steht als Abkürzung für COoling (Kühlung) und weist auf die Verfahren der Elektronenkühlung und der stochastischen Kühlung hin, die angewendet werden bzw. werden sollen, um einen stark gebündelten (kalten) Teilchenstrahl zu erhalten. SY steht als Abkürzung für SYnchrotron und bezeichnet die Methode der Teilchenbeschleunigung. Die Anpassung der Feldstärken der Magnete und die Veränderung der Beschleunigerfrequenz muß dynamisch und synchron erfolgen.

Die zukünftige COSY-Nutzergemeinschaft hat sich in der CANU (COSY-Arbeitsgemeinschaft Nordrhein-westfälischer Universitäten) organisiert. Die CANU und weitere internationale Kollaborationen haben bisher 20 COSY-Experimente vorgeschlagen.
Bild 1-1 zeigt den Aufbau des COSY-Jülich.

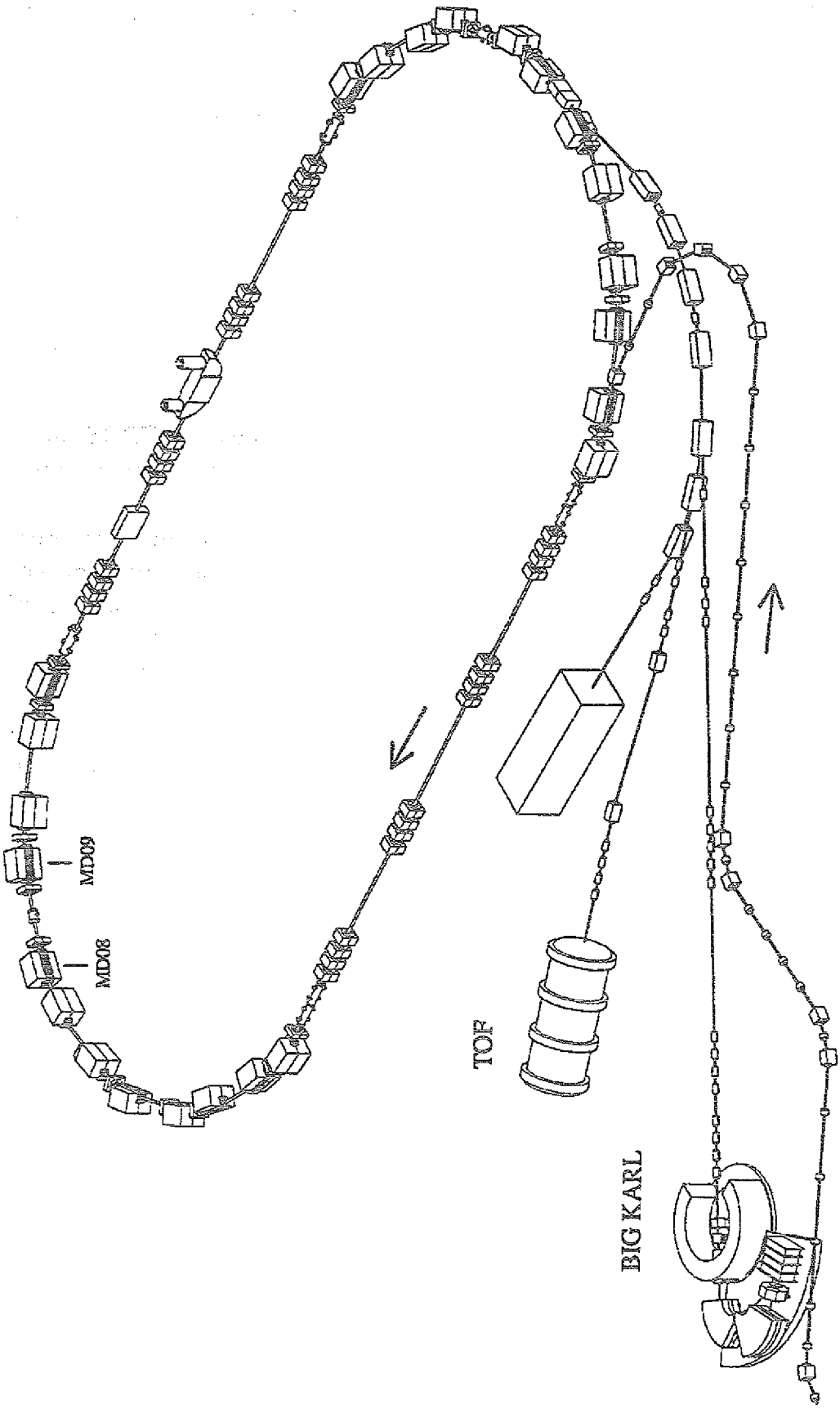


Bild 1-1: Das COSY-Jülich

Die Protonen oder Ionen werden vom Zyklotron des IKP (Institut für Kernphysik) aus, durch ein Strahltransportsystem (beam-line), in den COSY-Ring injiziert. Der Strahl bewegt sich durch eine Vakuumröhre in der ein Druck von etwa 10^{-10} mbar herrscht. Das ist ein geringerer Druck als im Weltall! Bei jedem Umlauf legen die Teilchen eine Strecke von etwa 184 m Länge zurück. Die Geradenstücke sind jeweils 40 m lang und die Halbkreise and den Enden haben einen Radius von 7 m. Jeder der 24 C-förmigen Dipolmagneten lenkt die Teilchen dabei um 15 Grad ab. Ein Dipolmagnet hat eine Eisenkernlänge von 1,83 m, einen Luftspalt von 9 cm, wiegt 27 Tonnen (!) und kann eine magnetische Induktion von maximal 1,58 Tesla erzeugen. Insgesamt 56 Quadrupolmagnete fokussieren den Strahl in der Vakuumröhre. Die Quadrupolmagnete haben eine Eisenkernlänge von 50 cm, eine Öffnung von 17 cm Durchmesser und wiegen zwischen 2- und 3 Tonnen. Sie arbeiten mit einer magnetischen Induktion von maximal 7,8 Tesla.
Die Extraktion des Strahls erfolgt nahe der Injektionsstelle und leitet den im Ring speicherbaren Strahl auf Wunsch zu externen Zielen. In Bild 1-1 sind der TOF-Detektor (TOF = Time-of-Flight) und das Massenspektrometer BIG KARL zu sehen.
Beim Betrieb von COSY sind aber auch Experimente vorgesehen, bei denen die Ziele direkt mit dem internen Strahl beschossen werden. Teile des Experiments sind dann innerhalb des COSY-Rings aufzubauen /LANG 88/.
Das in den Kapiteln 3, 4 und 5 beschriebene Experiment COSY 11 ist ein internes Experiment.

1.4.1. Was kann mit COSY erforscht werden?

COSY soll hauptsächlich dazu dienen, Antworten auf folgende Fragen zu erhalten:

1. Wie beeinflussen sich instabile Hadronen (Baryonen oder Mesonen) und Protonen, Neutronen oder andere Hadronen gegenseitig?
2. Gibt es außer den bekannten Baryonen und Mesonen noch andere?
3. Die Theorie sagt voraus, daß sich Gluonen als meßbare Partikelstrukturen, sogenannte Gluebälle, formieren können. Ist dies korrekt?
4. Ist die Quarkstruktur von Baryonen in einem Atomkern stabil oder verändert sie sich? (Daraus ergibt sich z. B. die Frage: Ist die Halbwertszeit eines Protons endlich oder unendlich?)

Der Arbeitsbereich von COSY soll Teilchenenergien von 40 MeV bis 2,5 GeV umfassen (MeV = Mega Elektronenvolt). Es sind also Experimente im Bereich der Atomphysik, Kernphysik und Mittelenergiephysik möglich. Mit anderen Worten: Man kann mit Teilchen experimentieren, die Up-, Down- und Charm-Quarks enthalten. Teilchen höherer Energiezustände können nur mit größeren

Beschleunigern wie z. B. am CERN (Organisation Européenne pour la Recherche Nucleaire) in Genf oder bei DESY (Deutsches Elektronen-Synchrotron) in Hamburg untersucht werden.

1.4.2. Datenerfassungssysteme am COSY-Jülich

Bild 1-2 zeigt die abstrahierte Struktur der Datenerfassungssysteme am COSY-Jülich.

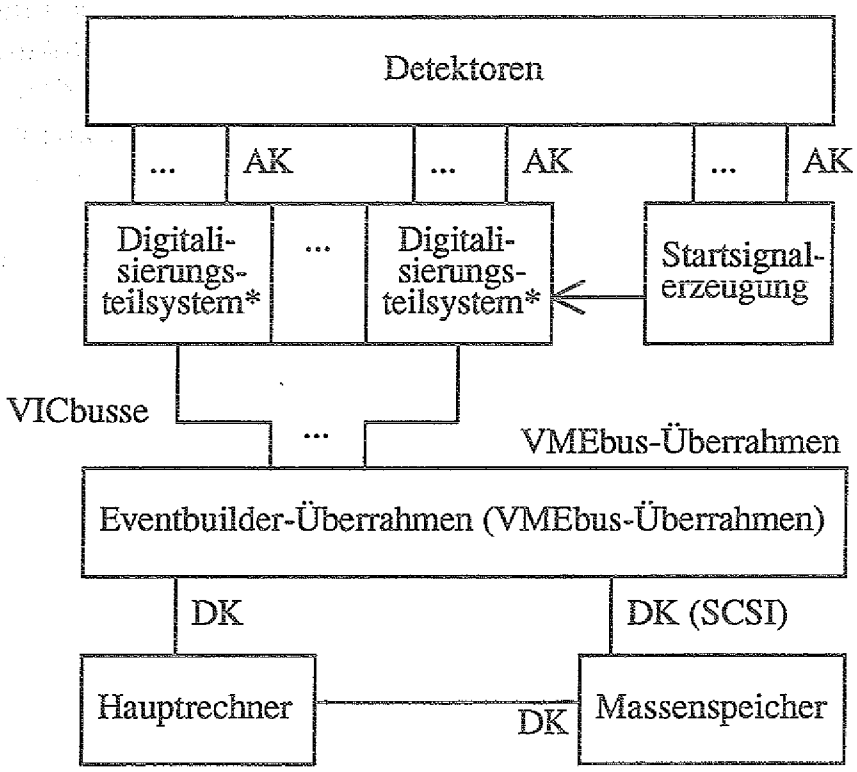


Bild 1-2: Struktur von Datenerfassungssystemen am COSY-Jülich

Eine Kollision eines Teilchens aus dem COSY-Ring mit einem Zielteilchen ist ein Ereignis, das über Detektoren elektrische Impulse in einigen analogen Meßkanälen (AK) erzeugt. Eine schnelle Elektronik trifft eine Vorauswahl, indem sie nur für bestimmte Ereignisse ein Startsignal für die Digitalisierungselektronik erzeugt. Erst dann sagt der Physiker: "Ein Kanal hat gefeuert".
Die Signale werden dann von Digitalisierungsteilsystemen in digitale Datenströme umgesetzt. Dazu werden Analog-Digitalumsetzer (ADCs) und Zeit-Digital-umsetzer (TDCs) verwendet.

Diese Systeme sind durch Module in CAMAC-, FASTBUS- und VMEbus-Überrahmen realisiert.
CAMAC (Computer Aided Measurement and Control) ist ein Standard zur Datenübertragung /ESON 83/.
FASTBUS ist ein Busstandard /ANSI 86/.
VMEbus (Versa Module Europe Bus) ist ebenfalls ein Busstandard /ANSI 87/.

Diese Überrahmen besitzen meist ein Controllermodul, welches die Daten blockweise in einem Speicher bereitstellt. Diese Daten werden über einen VICbus-Strang weitertransportiert.
VICbus (VME Inter-Crate Bus) ist ein Busstandard für die Datenübertragung zwischen mehreren Überrahmen /ISO 93/ /PARK 93/.

Ein VMEbus-Rahmen bildet die Wurzel der Baumstruktur des Datenerfassungssytems. Er ist mit je einem Modul für jeden VICbus-Zweig ausgestattet, welches die Datenblöcke eines Ereignisses zu einer logischen Untereinheit (Subevent) zusammenstellt. Ein solches Modul wird auch Subeventreader (SER) genannt.
Der Controller des Wurzel-Überrahmens akkumuliert die Untereinheiten zu einem logischen Datenblock: Dem Ereignis (Event). Dieser Controller wird Eventbuilder (EB, Ereignisbildner) genannt. Der gesamte Überrahmen wird auch als Eventbuilder-Überrahmen bezeichnet.
Ein Hauptrechner steuert und kontrolliert das gesamte System.
Die Events werden vom Eventbuilder über den SCSI-Bus (SCSI = Small Computer Systems Interface) auf ein schnelles Magnetband, zur späteren Bearbeitung, eigenständig übertragen.
Diese Struktur entspricht den Empfehlungen der CANU-Kommission und ist in /ZWOL 94/ beschrieben.

1.5. Motivation

Die hohe Strahlintensität und -qualität heutiger Beschleuniger führt zu hohen Kollisionsraten der Teilchen und damit zu hohen Datenraten am Detektor. Bild 1-3 gibt eine Übersicht der erwarteten Datenraten für einige Experimente am COSY.

Experiment Nr.	Startsignalrate in Hz	Feuernde Kanäle	Datenrate in kbyte/s
1	< 100	20	4
2	10^4	30	1400
3	50	256	25
6	10	40	2
8	10^4	40	400
9	400	20	20
10	$< 4 \times 10^3$	20	< 200
12	50	200	< 100
13	1	200	1
15	30	300	40

Bild 1-3: Erwartete Datenraten einiger COSY-Experimente /ZWOL 91/.

In Spalte drei von Bild 1-3 ist die mittlere Anzahl der feuernden Kanäle, für Ereignisse mit Startsignal, aufgelistet. Nur diese analogen Signale werden in digitale Werte umgesetzt und bilden ein Datenpaket, das Event (Ereignis) genannt wird. Die Meßdaten fallen also stochastisch in Form von Paketen an. Die mittlere Frequenz des Startsignals, und damit die Paketrate am Eingang des Systems, ist in Spalte zwei eingetragen. Mit dem Wissen, wieviel Byte Daten pro Meßkanal erzeugt werden, läßt sich die in Spalte vier aufgeschriebene Datenrate errechnen.
Die Anforderung an die elektronische Datenerfassungsanlage von COSY ist somit die On-Line-Erfassung von Daten, mit Raten im Bereich von 1- bis 1400 kbyte/s.
Das heute existierende System hat diese Fähigkeit /ZWOL 94/. Es ist jedoch abzusehen, daß dieses System für zukünftige Experimente an COSY eine zu enge Grenze bezüglich der Datenerfassungsrate setzt.
Diese Arbeit beschreibt ein optimiertes System das folgende Eigenschaften haben soll:

1. Höhere Datenerfassungsrate (> 1,25 Mbyte/s)
2. Optimiertes Softwarekonzept
3. Fähigkeit zur On-Line-Kompression

Erläuterung zu Punkt 3:

Durch die hohen Datenraten entstehen nach einiger Zeit auch große Datenmengen, deren spätere Auswertung entsprechend viel Zeit erfordert.
Um diese Zeit zu verkürzen kann man, wie bei der Datenkompression in der Bildübertragung, die zu speichernde Datenmenge reduzieren indem man sofort beim Entstehen der Daten durch intelligente Algorithmen dafür sorgt, daß nur "wichtige" Daten gespeichert werden. Geschieht dies ohne die Datenerfassungsrate zu reduzieren, spricht man von On-Line-Kompression. Es ist so auch möglich die Datenerfassungsrate zu steigern, da ja durch die Kompression weniger Daten pro Zeiteinheit gespeichert werden müssen.
Die Forderung 1 läßt sich also u. a. durch das Erfüllen der Forderung 3 realisieren.

1.6. Übersicht

In Kapitel 2 wird eine Literaturrecherche über Datenerfassungssysteme ähnlicher Experimente an anderen Beschleunigern durchgeführt, deren Ergebnis eine **typische Struktur** ist.
An einem in Kapitel 3 beschriebenen **konkreten Experiment** am COSY-Jülich wird der physikalische Hintergrund und der Aufwand für die Detektion von Teilchen erläutert. Daraus ergibt sich die in Kapitel 4 dargestellte Struktur der Datenerfassungsanlage für das Experiment COSY 11 als eine bereits **realisierte Lösung**. Damit ist der gegenwärtige Zustand (Ist-Zustand) des Systems beschrieben.

Um eine **optimierte Lösung** zu finden, wird in Kapitel 5 die Struktur von Experimenten an COSY verallgemeinert und mit Hilfe der Programmiersprache occam 2 zur Problembeschreibung dargestellt. Die dabei sichtbar werdenden Strukturschwächen werden in Kapitel 6 behoben und das **Konzept für die Implementierung** eines optimierten Systems wird vorgestellt.
In der anschließenden Gesamtzusammenfassung dieser Arbeit wird auch ein Ausblick auf noch bessere Realisierungsmöglichkeiten gegeben.
Im Anhang findet man, außer dem Glossar, dem Abkürzungs- und Literaturverzeichnis, auch einen Teil der technischen Dokumentation für das vorher beschriebene Implementierungskonzept. Dieser Teil der Arbeit soll einen Eindruck von der Komplexität der Hardware vermitteln.

Liest man die oben fett gedruckten Begriffe direkt nacheinander, ergibt sich der "rote Faden" für diese Arbeit.

2. Strukturanalyse von DV-Systemen der Hochenergiephysik

Die Struktur eines Datenerfassungssystems läßt sich in vier Hauptebenen gliedern (Bild 2-1). Die erste Ebene besteht aus Sensoren, die physikalische Größen in elektrische Signale umwandeln.
In der Ebene 2 werden aus diesen analogen Meßsignalen digitale Datenströme erzeugt. Diese Ebene wird Digitalisierungsebene oder Frontend genannt. Durch eine Elektronik werden Startsignale (Trigger) erzeugt, die die Digitalisierung steuern. Über mehrere digitale Kanäle (Busse) werden die Daten, nach Vorverarbeitung in Ebene 3, einem Massenspeicher in Ebene 4 zugeführt. Das System wird durch einen Hauptrechner (Host) gesteuert und überwacht.

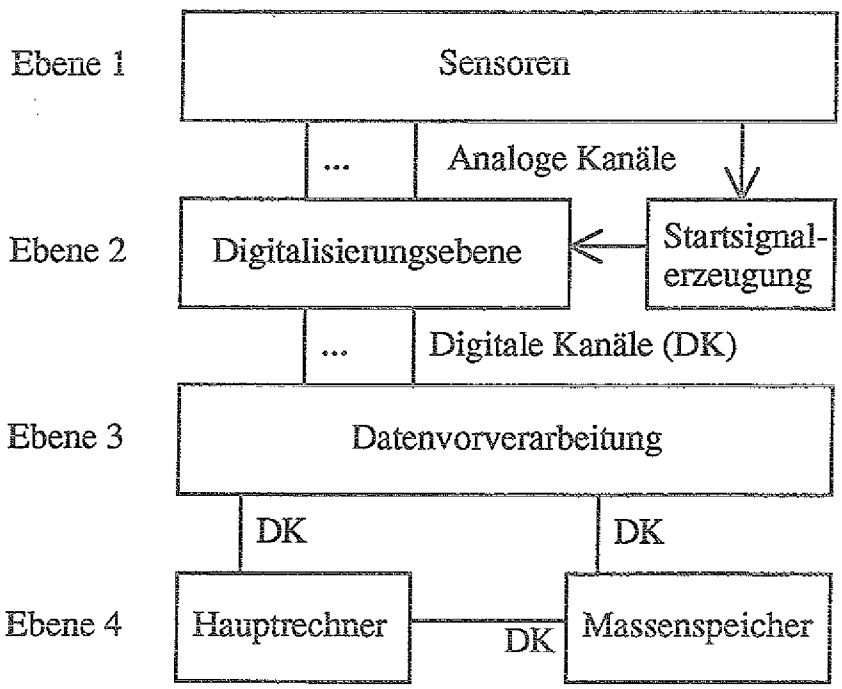


Bild 2-1: Die Ebenen eines Datenerfassungssystems

Die in der Hochenergiephysik verwendeten Strukturen zur Datenerfassung sind in der Literatur beschrieben. Die Bilder 2-2 bis 2-5 geben einen Überblick. Als Quellen wurden hauptsächlich die "IEEE Proceedings of Nuclear Science", sowie die Konferenzberichte: "1991 IEEE Nuclear Science Symposium, Santa Fe, USA", "IEEE Seventh Conference REAL TIME 91, Jülich, Germany" und "Open Bus Systems 92 in Research and Industry, Proceedings, Zürich, Switzerland" verwendet. Es wird somit kein Anspruch auf Vollständigkeit erhoben. Die Bilder 2-2 bis 2-5 geben die Literaturstellen und deren Häufigkeit für eine bestimmte Systemart wieder, so daß man auf den ersten Blick oft benutzte Lösungen erkennt und sofort einige Referenzen hat.

Diese Methode habe ich aus folgendem Grund gewählt: Eine Randbedingung in der modernen Forschung ist, daß bei der Ausrüstung von neuen Experimenten möglichst auf bereits vorhandene Systeme zurückgegriffen wird. Dies geschieht zur Verkürzung der Entwicklungszeit und aus Kostengründen. Dabei ist es sinnvoll, Lösungen zu verwenden oder weiter zu entwickeln, die von vielen Benutzern erfolgreich verwendet wurden oder die sogar standardisiert sind.

Bild 2-2 zeigt eine Übersicht der Literaturstellen sortiert nach den unterschiedlichen Bussystemen für die Digitalisierungsebene.
CAMAC /ESON 83/, FASTBUS /ANSI 86/, VME /ANSI 87/ und VXI sind Standards für Busse, die durchaus vergleichbar sind mit den bekannteren Bussen, die man in PCs findet, wie etwa: ISA (Industrial Standard Architecture), EISA (Extended Industrial Standard Architecture) oder den Microchannel.

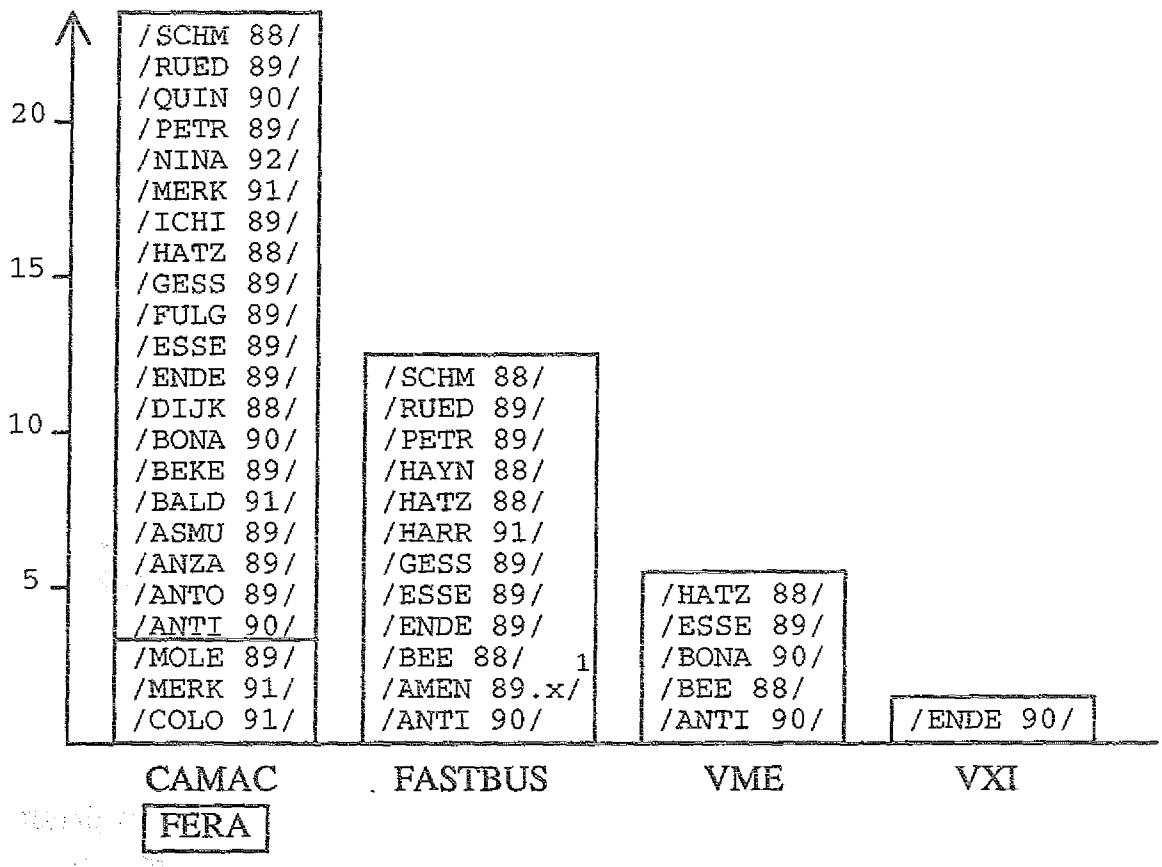


Bild 2-2: Die Häufigkeiten der Literaturstellen von in der Hochenergiephysik verwendeten Digitalisierungs-Bussystemen.[1]

[1]/AMEN 89.x/ bedeutet, daß von diesem Autor im gleichen Jahr mehrere Veröffentlichungen erschienen sind.

Bei den auf der Digitalisierungsebene eingesetzten Systemen in Bild 2-2 erkennt man, daß hier meist CAMAC- oder FASTBUS-Systeme eingesetzt werden. CAMAC ist zwar eine recht alte Norm (1969) jedoch sind dafür viele Module, mit hoher Güte für Meßaufgaben der Physik, von mehreren Anbietern erhältlich. Der Einsatz von FASTBUS ist meist durch höhere Geschwindigkeitsanforderungen oder durch das Vorhandensein einer großen Anzahl von Meßkanälen begründet. Bei COSY werden, außer der VXI Norm, alle aufgelisteten Standards verwendet. Die Komponenten der Firma FERA haben den Vorteil, daß sie CAMAC-Systeme durch einen zusätzlichen Datenbus kostengünstig schneller machen können. Das VME-System wird eingesetzt, weil es auch in Ebene 3 der Datenerfassungshierarchie verwendet wird.
Die Verbindung zwischen den Ebenen 2 und 3 eines Datenerfassungssystems nennt man im Bereich der Hochenergiephysik auch Instrumentierungsbus /PONT 91/. Bild 2-3 zeigt die Häufigkeiten der Literaturstellen der verwendeten Instrumentierungsbusse.

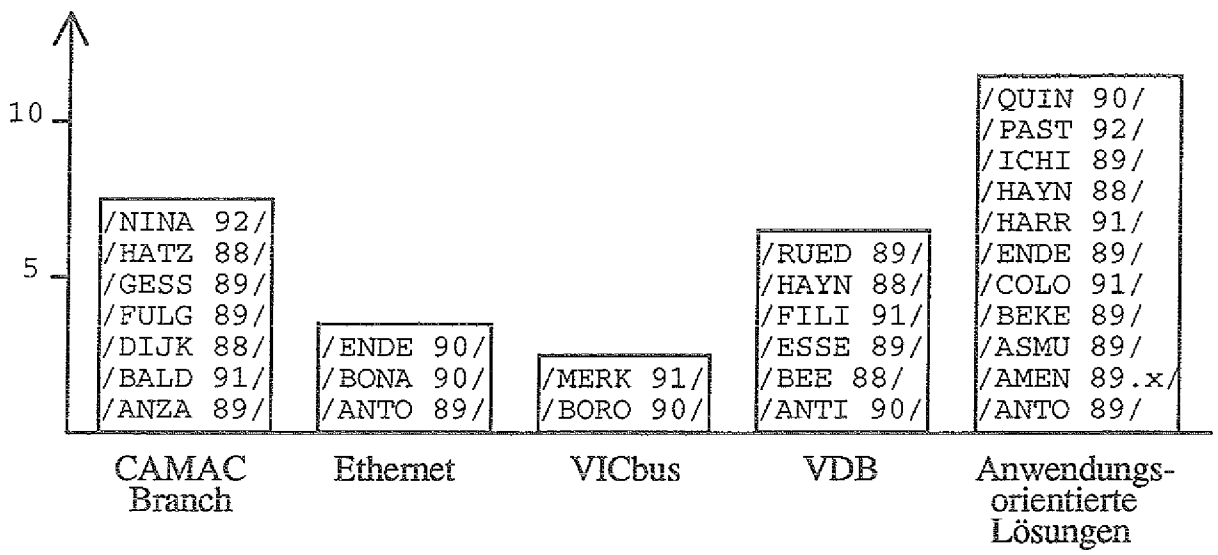


Bild 2-3: Die Häufigkeiten der Literaturstellen von in der Hochenergiephysik verwendeten Instrumentierungsbussen.

Bei der Übersicht der verwendeten Instrumentierungsbusse erkennt man deutlich einen Schwerpunkt bei den anwendungsorientierten Lösungen, d. h. viele Arbeitsgruppen haben nicht Ergebnisse anderer wiederverwendet, sondern eigene Entwicklungsarbeit geleistet. Die Gründe dafür sind die geringen Datentransferraten der damals verfügbaren Standards: CAMAC Branch (CAMAC Zweig) und Ethernet, sowie die Tatsache, daß der VDB kein Standard für Inter-Überrahmenverbindungen ist. Der VDB ist lediglich eine Erweiterung des VSB (VME Subsystem Bus), die differentielle Treiber benutzt, um größere Leitungslängen zu erreichen.

Damit in diesem Bereich Zeit und Kosten gespart werden können, mußte also ein neuer Standard mit einer hohen Transferrate geschaffen werden: Der VICbus. Er ist in /ISO 93/ und /PARK 93/ beschrieben und wird bei COSY verwendet. Es finden sich hier noch wenige Anwendungen mit dem VICbus, da einige Entwicklungen dazu erst später erfolgten, oder noch im Gange sind /ERVE 92/ /HAGE 93/ /HOLZ 92.1/ /HOLZ 92.2/ /STRU 92/.

Bild 2-4 zeigt die Häufigkeiten der Literaturstellen der verwendeten Systeme für die Datenvorverarbeitung. Um den Datenstrom vom Instrumentierungsbus auf einen Massenspeicher zu leiten, bedarf es einer Datenvorverarbeitung, die - im einfachsten Fall - von einem Mikroprozessor mit Speicher und entsprechenden Schnittstellen geleistet werden kann. Ein System aus diesen Komponenten wird auch Knoten (Node) genannt.
In der Regel benötigt man aber ein Multiprozessorsystem mit geeigneten Kommunikationskanälen, also wieder einen Systembus oder auch die Punkt-zu-Punkt Verbindungen eines Transputersystems.

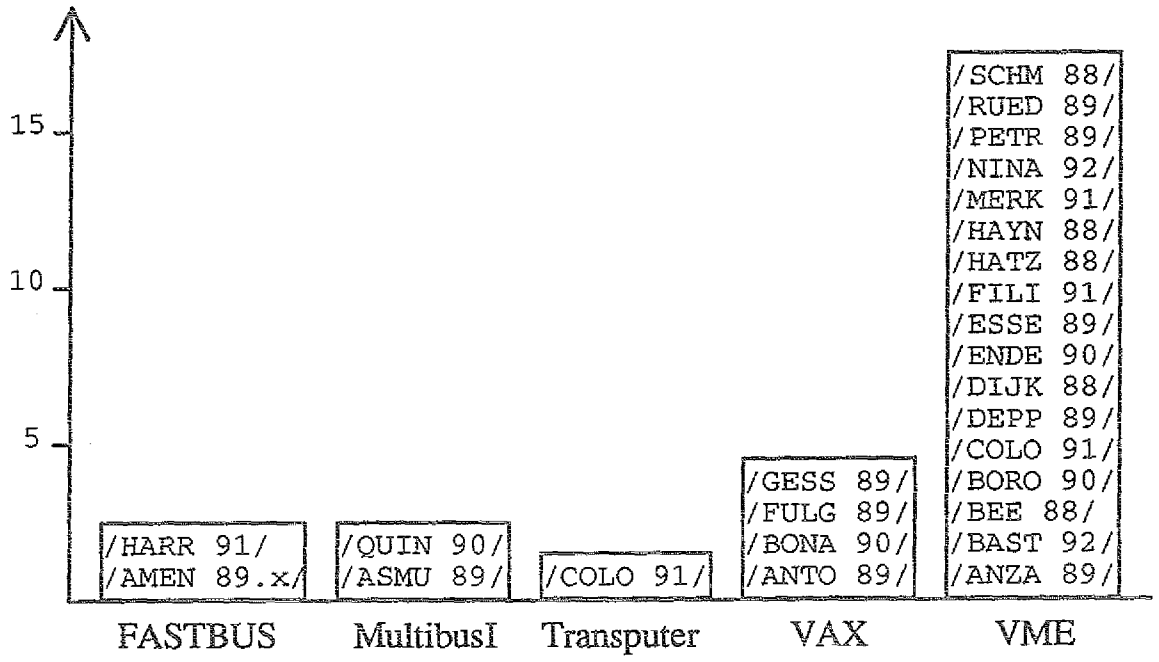


Bild 2-4: Die Häufigkeiten der Literaturstellen der verwendeten Systeme für die Datenvorverarbeitung

Bei den Datenvorverarbeitungssystemen liegt der Schwerpunkt deutlich bei den VMEbus-Systemen, denn hier wird viel Rechenleistung verlangt. Da für den VMEbus eine Vielzahl von Mikroprozessor-Modulen mit großer Leistung käuflich ist, erklärt sich so die Auswahl der Anwender. Bei COSY ist an dieser Stelle ebenfalls der Einsatz von VMEbus-Systemen geplant.

Bild 2-5 zeigt eine Aufstellung der Literaturstellen der verwendeten Hauptrechner (Hosts). Eine Aufgabe des Hauptrechners ist die Steuerung (Leitrechnerfunktion) der Datenerfassung. Der Hauptrechner dient auch als Monitor für die Überwachung der Datenübertragung und stellt die Mensch-Maschine-Schnittstelle zur Verfügung. Eine weitere Aufgabe des Hauptrechners kann die Datenauswertung und -darstellung sein. Dazu ist ein solcher Rechner, wie bei COSY vorgesehen, mit den nötigen Betriebsmitteln auszustatten.

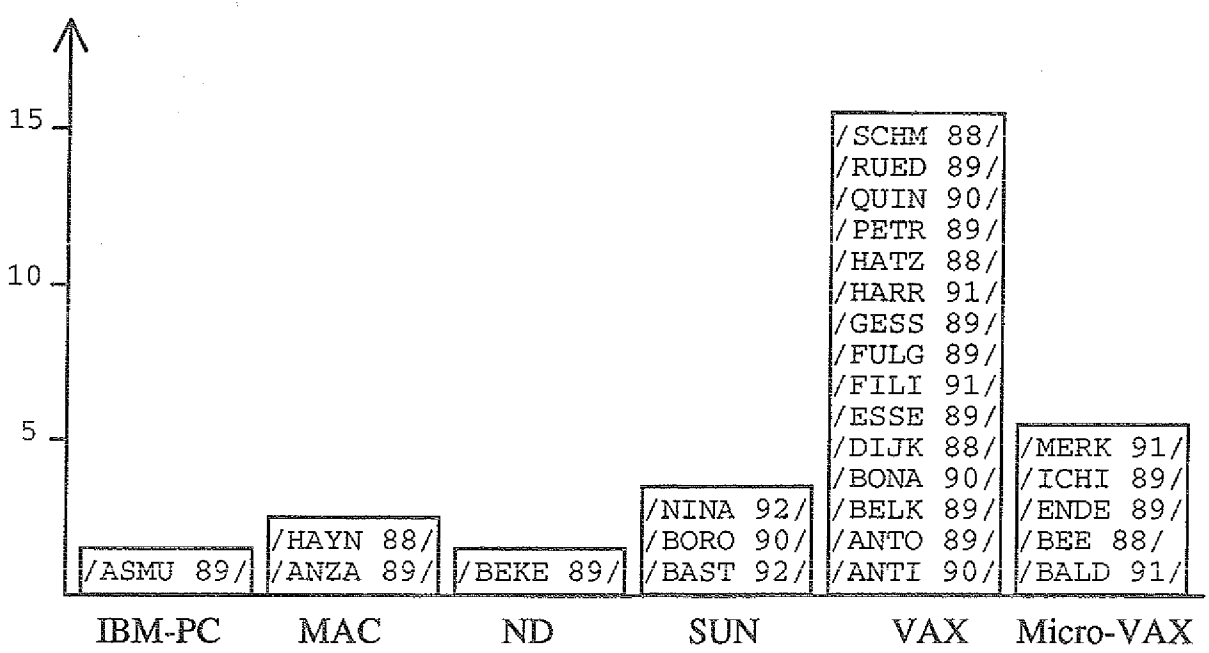


Bild 2-5: Die Häufigkeiten der Literaturstellen der bei Hochenergiephysik-Experimenten verwendeten Hauptrechner

Bei den verwendeten Experimentsteuerrechnern sieht man, daß hier meist VAX-Rechner eingesetzt wurden. Das ist wohl durch die Verfügbarkeit, Sicherheit und den Preis dieser Rechner, zum Zeitpunkt der Entwicklungen, zu erklären.

Das Ergebnis dieser Literaturrecherche ist, daß sich die Struktur der meisten Systeme wie in Bild 2-6 verallgemeinert darstellen läßt.

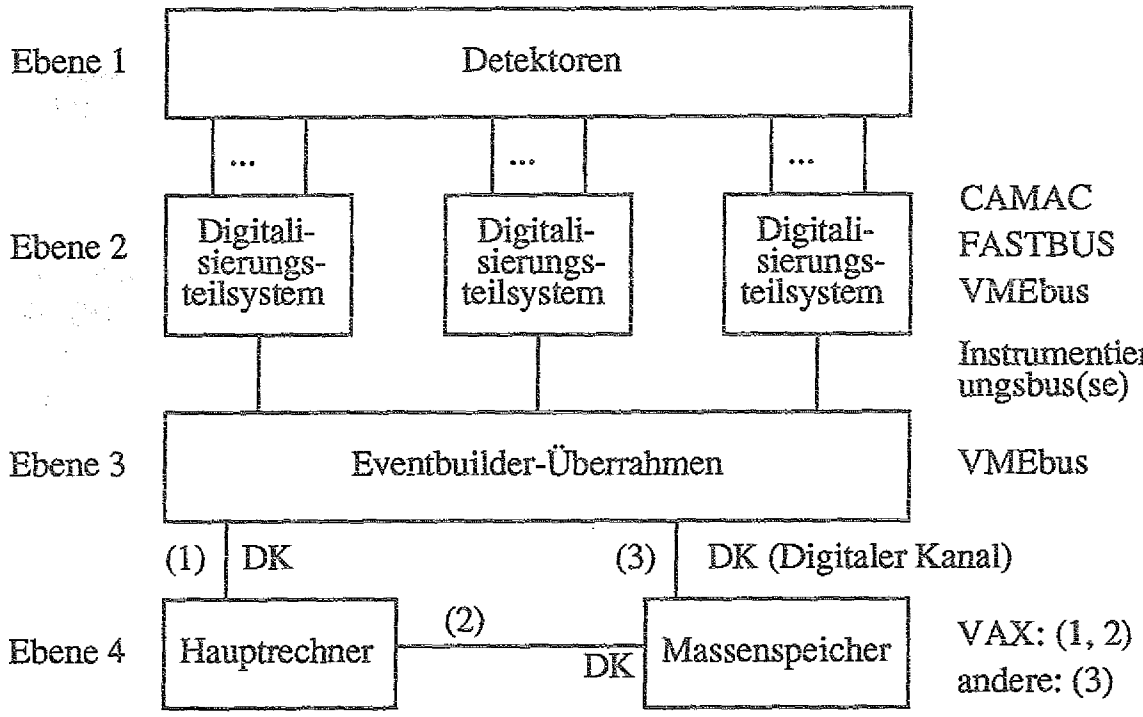


Bild 2-6: Struktur von Datenerfassungssystemen in der Hochenergiephysik

Die erste Ebene der Detektoren (Sensoren) ist je nach Experiment verschieden. Hier lassen sich keine Angaben wie: "Häufig benutzte Detektoren sind ... " machen.
Auf der zweiten Ebene befindet sich die Frontend- oder Digitalisierungselektronik. Hier werden meist die Standards: CAMAC, FASTBUS und VMEbus verwendet. Die eingesetzte Elektronik zur Startsignalerzeugung (Trigger-Generation) ist je nach Experiment anders und hier nicht dargestellt.
Vor der dritten Ebene werden die Digitalisierungsteilsysteme durch Instrumentierungsbusse mit der Datenvorverarbeitung verbunden. Im Bereich der Hochenergiephysik wird die Ebene 3 als Eventbuilder (Ereignisbildner) bezeichnet. Es werden die verschiedensten Instrumentierungsbusse eingesetzt. Der VICbus wird wohl erst in Zukunft häufiger eingesetzt werden. Als Eventbuilder werden meist VMEbus-Systeme gewählt.
Die vierte, unterste Ebene, wird durch den Haupt- oder Steuerrechner gebildet. Bei der Verbindung (1) zum Steuerrechner wird meist Ethernet verwendet. Es werden dann die verschiedensten Rechner zur Experimentsteuerung eingesetzt. Heute sind das oft DEC-, SUN- und Micro-VAX-Rechner, aber auch PCs von Macintosh und IBM.

Der Massenspeicher ist oft ein schnelles Magnetband, welches die maximale unkomprimierte Datendurchsatzrate auf z. Z. etwa 1,25 Mbyte/s begrenzt /FANO 89/ /DEC 94/. Ein Ansatz mehrere Magnetbänder parallel einzusetzen findet sich in /ROST 91/. Diese Architektur ist jedoch einmalig.
Es gibt nun zwei Möglichkeiten, das Magnetband an das System zu koppeln. Früher war der Hauptrechner meist eine VAX, mit dem echtzeitfähigen Betriebssystem VMS, welches das Magnetband über die Verbindung (2) direkt bedienen konnte. Der moderne Lösungsansatz ist aber das Echtzeitbetriebssystem OS-9 auf der Ebene 3 in CPUs (Central Processing Units) innerhalb von VMEbus-Überrahmen zu verwenden, um von hier aus das Magnetband über die Verbindung (3) zu bedienen. Das hat den wesentlichen Vorteil, daß der Hauptrechner mit einem Standard-UNIX Betriebssystem auskommt, das nicht echtzeitfähig ist. So kann eine große Palette von Standard-Software genutzt werden. Für das verarbeiten und visualisieren der Daten wird z. B. das Softwarepaket PAW (Physics Analysis Workstation) benutzt /JOHN 89/.

2.1. Zusammenfassung

Datenerfassungssysteme lassen sich generell in vier Ebenen gliedern.
Das Ergebnis der Literaturrecherche ist, daß sich die Strukturen der Datenerfassungssysteme im Bereich der Hochenergiephysik noch verfeinert - und dennoch verallgemeinert - darstellen lassen. Die Digitalisierungsebene gliedert sich in Digitalisierungsteilsysteme, die über Instrumentierungsbusse sternförmig mit dem Eventbuilder verbunden sind. Für die Ebenen der Digitalisierung und des Eventbuilders habe ich typische Methoden der Realisierung gefunden.
Diese Ergebnisse verifizieren, daß die in den Kapiteln 3, 4 und 5 beschriebene Struktur der Datenerfassungsanlage für das Experiment COSY 11 und die Struktur des optimierten Systems (Kapitel 6) sinnvolle Lösungen sind.

3. Die Physik des Experiments COSY 11

Die hier beschriebene Aktivität liefert einen Beitrag zur Instrumentierung von Experiment Nr. 11: "Threshold Meson Production at the Internal COSY-Beam in the Range of Scalar Mesons Involving Strangeness" (In deutsch etwa: Meson Produktion an der Produktionsschwelle am internen COSY-Strahl im Bereich skalarer Mesonen, die Strange-Quarks enthalten). Der verantwortliche Kollaborationssprecher für dieses Experiment ist Herr Dr. W. Oelert, ein Mitarbeiter des Forschungszentrums Jülich im IKP (Institut für Kernphysik) /OELE 92/.
Für alle COSY-Experimente wird eine modulare und standardisierte Datenerfassungslösung angestrebt, d. h. die in dieser Arbeit erzielten Ergebnisse für das Experiment COSY 11 werden auch auf andere Experimente übertragbar sein.
Der Aufbau von Experiment Nr. 11 ist in den Bildern 3-1 und 3-2 skizziert. Man erkennt in Bild 3-1 einen Ausschnitt aus dem internen Strahlverlauf von COSY 11 (vgl. Bild 1-1). Die Vakuumröhre ist hier nicht dargestellt. Die Teilchen bewegen sich im Magnetfeld der Dipole (z. B. MD09 (Magnetischer Dipol Nr. 9)) auf einer kreisförmigen Bahn und außerhalb der Dipole geradlinig fort.
Mit diesem Experimentaufbau wird der Dipolmagnet MD08, in erster Linie Bestandteil der Elektronenoptik des internen Strahls, als Magnet für ein Massenspektrometer genutzt.

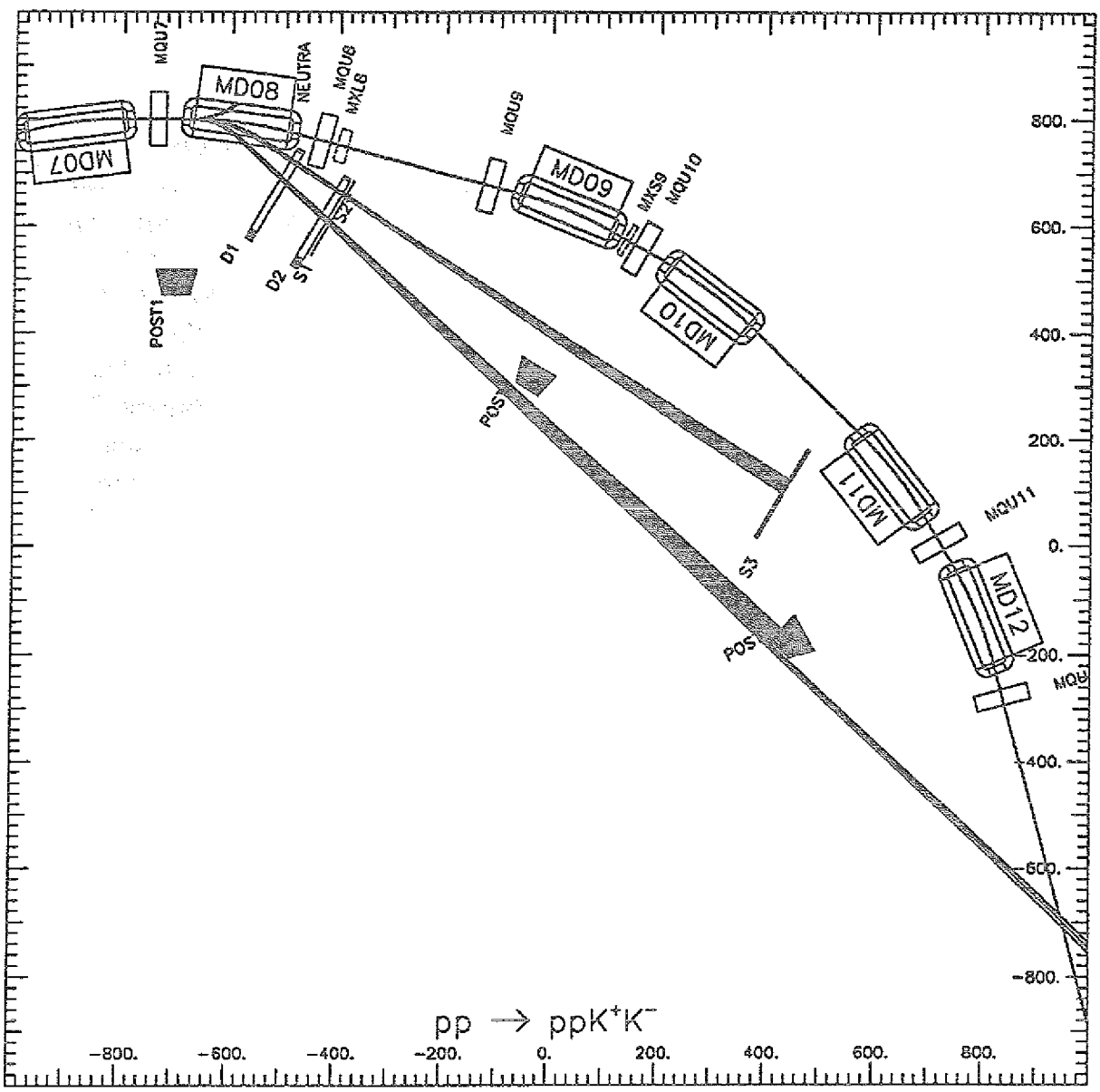


Bild 3-1: Schematischer Aufbau des COSY Experiments Nr. 11 mit Teilchenbahnen für die Reaktion: pp -> ppK$^+$K$^-$, mit Zentimeter-Skalen

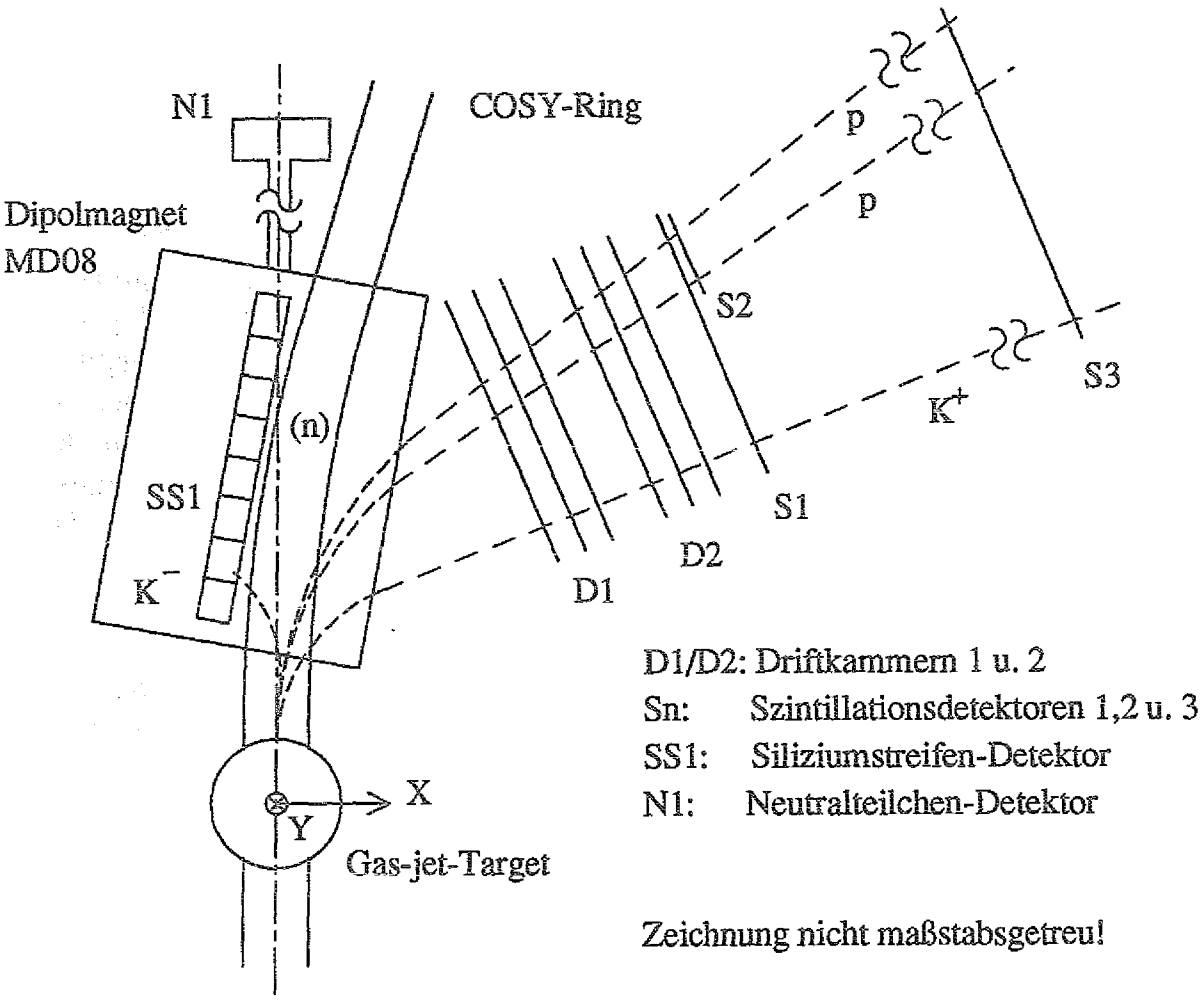


Bild 3-2: Schematischer Aufbau des COSY Experiments Nr. 11 mit Teilchenbahnen für die Reaktion: pp -> ppK$^+$K$^-$ (Zusammenstellung aus Bild 3-1)

Es werden Produktionsreaktionen vom Typ pp > ppX im Bereich der Produktionsschwelle studiert z. B. pp > ppX ; X > K K. Die Terminologie: "pp > ppX" besagt, ein im COSY beschleunigtes p (Proton) macht Wechselwirkung mit einem weiteren p, das ein sogenanntes Gas-jet-Target (Gasstrahlziel) liefert. Dabei wird das System ppX erzeugt, das weiter zerfällt. Die Abkürzung: K$^+$ steht für ein positiv geladenes Kaon (ein Meson).
In diesem Experiment sind die verschiedensten Teilchenparameter zu messen, z. B. Teilchendurchflugsposition und Teilchendurchflugszeit, Energieverlust und dE/dx (Energieverlust pro Längeneinheit) u. s. w..

Die Teilchenparameter werden beim COSY Experiment Nr. 11 mit folgenden Detektortypen gemessen:

D1/D2: Zwei Driftkammern, die jeweils drei Ebenen mit verschieden ausgerichteten Drähten in Form von Gittern enthalten, befinden sich außerhalb des Dipols (Bild 3-2) und liefern Daten über den Spurverlauf der positiv geladenen Ejectile (herausfliegende Teilchen, z. B. nach einer Kollision).
Die Driftkammern sind mit einem Gas gefüllt, das beim Durchflug eines Teilchens primär ionisiert wird. Danach befinden sich freie Elektronen und positiv geladene Ionen in der Kammer. Die Elektronen bewegen sich nun, durch das elektrische Feld, zum nächstgelegenen Anodendraht hin. Die dafür benötigte Zeit wird Driftzeit genannt. Nahe der Oberfläche des Drahtes ist die Feldstärke so hoch, daß ein Elektron durch Stoßionisation vervielfacht wird und so im Draht einen meßbaren elektrischen Impuls erzeugt. Die Nummer des angesprochenen Drahtes liefert dadurch eine grobe Ortsinformation.
Nach dem Passieren der Driftkammern sendet der Detektor S1 einen Startimpuls aus, so daß die Zeit zwischen diesem Startimpuls und dem Eintreffen der Elektronen an einem Anodendraht durch einen TDC (Time to Digital Converter) gemessen werden kann. Diese Zeitspanne ist abhängig vom Abstand des Teilchendurchflugpunktes zum Anodendraht, und liefert so eine genauere Ortsinformation.
Die zur Kathode driftenden Ionen bewegen sich relativ langsam und sind für Messungen mit hoher Zeitauflösung unbrauchbar.
Für beide Driftkammern sind etwa 400 Meßkanäle vorgesehen.

S1: Der Silizium-Pad-Detektor S1 (Siliziumsperrschicht-Detektor) ist ortsauflösend und besitzt etwa 1000 Segmente. Er wird, wie für D1/D2 beschrieben, zur Auslösung der Driftzeitmessungen von D1 und D2 benutzt.

S2: Der Silizium-Pad-Detektor S2 (Siliziumsperrschicht-Detektor) verbessert die Auflösung von S1 in einer Zone, wo zwei Teilchenstrahlen aus Protonen dicht beieinander verlaufen. Die Detektoren D1, D2, S1 und S2 liefern somit die zur Teilchenspurberechnung notwendigen Informationen.

S3: Die Szintillationszählerwand besteht aus ortsauflösenden Szintillatorsegmenten in einer Entfernung von etwa 10 m hinter S1 und besitzt eine sehr gute Zeitauflösung

N1: An diesem Platz werden bei der Experimentdurchführung ein Kalorimeter-Detektor zur Energiemessung, und Flugzeitdetektoren zur Erfassung der erzeugten ladungsneutralen Teilchen aufgestellt.

Die Bezeichnungen D1, D2 usw. sind keine in /OELE 92/ festgelegten Namen. Sie sollen hier nur als Bezug zu Bild 3-1 und Bild 3-2 dienen.

Der zu untersuchende Produktionskanal läuft immer in Konkurrenz zu einer Vielzahl anderer Prozesse ab, man findet u. U. erst unter 1 Million Teilchen ein Teilchen aus dem gesuchten Prozeß. Deshalb muß man bei jedem Experiment für einen spezifischen Auslöser erster Ordnung (first-order trigger) sorgen, d. h. man baut sich eine Elektronik, die sehr schnell einige Bedingungen abfragt (z. B. koinzidente Teilchenzahl, Teilchenart, Multiplizität), die ein gewünschtes Ereignis erwarten lassen, und die dann mit einem Auslese-Startsignal das Lesen aller Detektoren nur für diese interessierenden Ereignisse startet.
Für den SS1 Detektor braucht man ein weiteres Signal, den Auslöser zweiter Ordnung (second-level trigger), der aus dem ersten Startsignal und den Daten der anderen Detektoren abgeleitet wird. Das Problem der Startzeitpunktsfestsetzung wird hier nicht weiter erläutert. Es ist aber festzuhalten, daß erst der Auslöser zweiter Ordnung die Datenabfrage im SS1 Detektor in Gang setzt.

SS1: Ein Siliziumstreifen-Detektor im Luftspalt des Dipols wird verwendet, um die negativ geladenen Teilchen, z. B. K^- (Kaon mit negativer Ladung), zu lokalisieren. Der SS1 liefert nacheinander die zu jedem Kanal gehörende Impulshöhe. Mit zwei besonderen Analog-Digitalumsetzern (Flash-ADCs) werden die Analogwerte dann digitalisiert. Diese Werte sind ein Maß für den Energieverlust pro Längeneinheit (dE/dx) und erlauben die Identifikation der Teilchen.

Die Signale der Elektronik von den verschiedenen Detektoren münden in Überrahmen, die mit ADCs (Analog to Digital Converters) oder TDCs (Time to Digital Converters) ausgerüstet sind. Wie die Architektur des Datenerfassungssytems für COSY 11 beschaffen ist, wird im nächsten Kapitel beschrieben.

3.1. Zusammenfassung

Das Experiment COSY 11 nutzt den internen Strahl des COSY-Jülich. Man erforscht hier die Mesonenproduktion, nach einer Proton-Proton-Kollision, an der Energieschwelle ihrer Entstehung. Dieser Energiebereich, bei dieser Reaktion, wurde bisher nicht genau untersucht. Für den Experimentaufbau sind aufwendige Detektoren mit einer großen Zahl von Meßkanälen notwendig. Daraus folgt, daß eine komplexe Datenerfassungsanlage eingesetzt werden muß. Diese wird im nächsten Kapitel beschrieben.

4. Architektur des Datenerfassungssystems für COSY 11

4.1. Übersicht

Bild 4-1 vermittelt einen Überblick zur Architektur des COSY 11 Datenerfassungssystems und zeigt die verwendeten Komponenten.

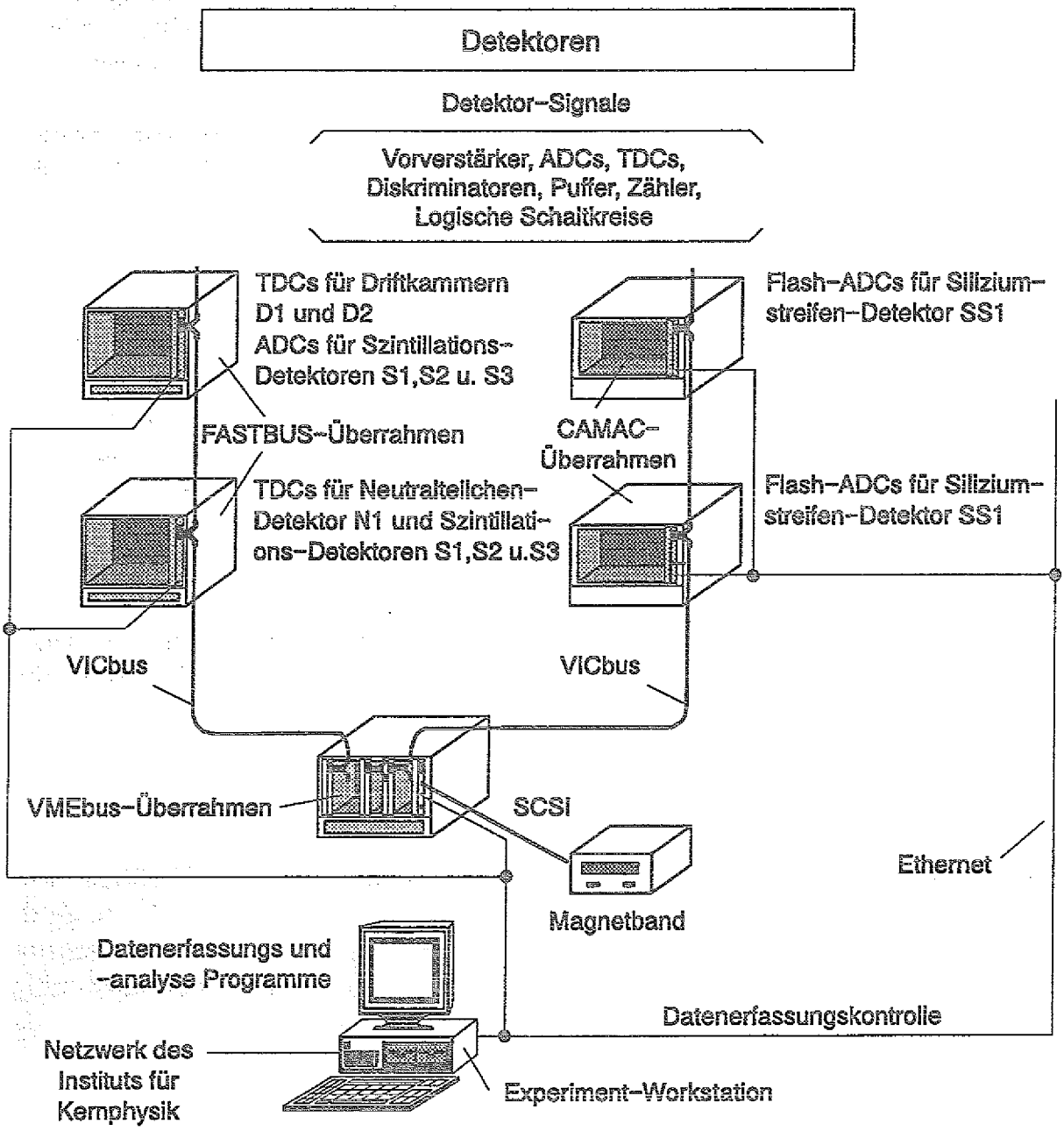


Bild 4-1: Aufbau des Datenerfassungssystems für das COSY Experiment Nr. 11

4.2. Digitalisierungskomponenten

Wie in Bild 4-1 zu sehen, werden für die Zeitmessungen mit den Detektoren N1, D1, D2, S1, S2 und S3 FASTBUS-Überrahmen verwendet. Für die Pulshöhenmessungen mit ADCs am Detektor SS1 werden CAMAC-Überrahmen eingesetzt. Eine nähere Erläuterung zu diesen Komponenten folgt in den nächsten Unterkapiteln.

4.2.1. CAMAC-Überrahmen

Ein CAMAC-Überrahmen (-Crate) ist mit einem Datenbus nach der CAMAC-Norm (CAMAC = Computer Aided Measurement and Control) ausgestattet /ESON 83/.
Die interne Datentransferrate beträgt bis zu 3 Mbyte/s. In einem CAMAC-Überrahmen wird ein Controller benötigt, der die Zugriffe der Module auf den gemeinsamen Bus verwaltet. Außerdem besitzt der Controller eine Schnittstelle als Verbindung zur Außenwelt. Im Zentrallabor für Elektronik ist ein intelligenter Überrahmen-Controller, mit Motorola 68030 CPU und VICbus Slave-Schnittstelle, entwickelt worden /ERVE 92/. Diese Entwicklungsarbeit erfolgte in Zusammenarbeit mit der Firma CES (Creative Electronic Systems) mit Hauptsitz in Genf.
Die CPU dieses Controllers verpackt die von den Modulen erfaßten Daten in Pakete, und legt sie in einem Zweitorspeicher (Dual-Port-RAM) ab. Bei Fertigstellung eines Pakets kann die steuernde Zentraleinheit (Master-CPU) am VICbus durch ein Signal (Interrupt) verständigt werden. Zur Reduzierung des Datenaufkommens ist eine Unterdrückung der Übertragung des Datums Null möglich.
Die Datenpakete können dann von der steuernden Zentraleinheit im Blocktransfer aus dem Zweitorspeicher geholt werden, während die Controller-CPU, parallel dazu, wieder ein neues Paket erstellt.

4.2.2. FASTBUS-Überrahmen

Ein FASTBUS-Überrahmen (-Crate), ist mit einem Bus nach der FASTBUS-Norm ausgestattet /ANSI 86/. Die Signale werden in ECL-Technik (ECL = Emitter Coupled Logic) realisiert, wodurch eine maximale Datentransferrate von 160 Mbyte/s erreicht werden kann.
Eine Besonderheit bei FASTBUS sind die extrem großen Leiterplattenabmessungen der Module: 366,7 mm Höhe und 400 mm Tiefe. Dadurch kann ein Modul an sehr viele Kanäle (typisch < 100) angeschlossen werden, und die Kosten pro Kanal sind dann geringer als bei CAMAC Modulen.
FASTBUS wird also vorzugsweise bei Detektoren mit großer Kanalzahl eingesetzt.

Um FASTBUS-Überrahmen in dieses Datenerfassungssystem integrieren zu können, war es notwendig eine Schnittstelle für einen FASTBUS-Controller zum VICbus zu entwickeln /STRU 92/.

4.3. Eventbuilder-Überrahmen

Das Herz des Datenerfassungssytems in Bild 4-1 ist ein VME-Überrahmen mit theoretisch bis zu 40 Mbyte/s interner Datentransferrate /ANSI 87/.
Für jeden VICbus-Zweig befindet sich hier eine CPU-Karte (Karte = Leiterplatte, Platine, gedruckte Schaltung) mit VICbus-Master Schnittstelle. Vom VICbus kommende Datenpakete müssen von diesen VME-CPUs schnell genug gelesen und abgespeichert werden.
Die Haupt-VME-CPU ist nicht direkt mit einem Digitalisierungsüberrahmen verbunden, sondern hat die Aufgabe, die von den anderen VME-CPUs gelieferten Datenpakete formatiert zu einem Datenpaket zusammenzustellen. Dieses Paket wird Event (Ereignis) genannt. In dieser Nomenklatur heißen dann die an den VICbussen angeschlossenen VME-CPUs: Subeventreader (Unterereignisleser), die Haupt VME-CPU: Eventbuilder (Ereignisbildner) und der ganze VME-Überrahmen: Eventbuilder-Überrahmen.
Die Events werden dann von der Haupt-VME-CPU über eine SCSI-Schnittstelle in Echtzeit auf ein schnelles Magnetband geschrieben. Hier werden die Experimentdaten bis zur späteren Auswertung gespeichert.

4.4. Datenflußstruktur des Datenerfassungssystems

Die Datenflußstruktur nach Bild 4-2 spiegelt den Hardwareaufbau nach Bild 4-1 wieder.

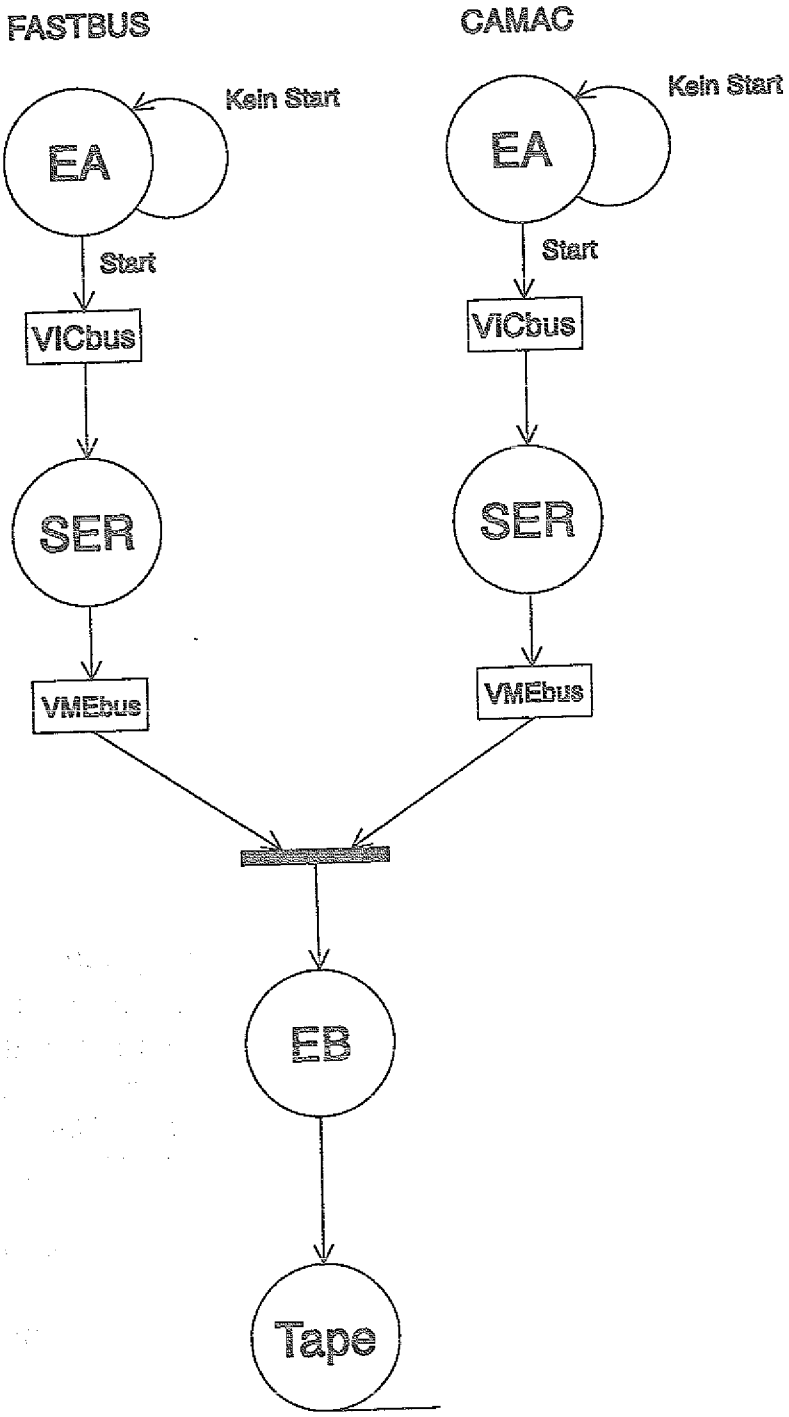


Bild 4-2: Datenflußdiagramm des Datenerfassungssystems nach Bild 4-1

Die beiden Zweige zur Datenerfassung, realisiert mit Hilfe von FASTBUS bzw. CAMAC, sind zu erkennen. Ein Event-Acquisitionsprozeß (EA) sammelt, nach dem Erkennen einer gültigen Startbedingung, die von den Umsetzern bereitgestellten Daten über das jeweils benutzte Bussystem in einen Wechselpufferspeicher. Ein EA-Prozeß läuft in der CPU eines Überrahmen-Controllers ab.
Die Daten stehen dann zum Transport über den VICbus bereit. Dies ist durch ein Rechteck in der Skizze dargestellt, das eine zeitliche Verzögerung symbolisiert.
Die Datenströme erreichen dann die Subeventreader-Prozesse (SER), die Formatierungsinformationen und anderes hinzufügen und die Datenpakete wieder in einem Wechselpufferspeicher bereitstellen.
Über den VMEbus werden die Daten dann zum Eventbuilderprozeß (EB) transportiert. Der EB-Prozeß bewirkt die Zusammenführung der Datenpakete zu einer zeitlich koordinierten und logischen Einheit (= Event), und legt die Daten wieder in einem Puffer ab. Der Tape-Prozeß (Magnetband-Prozeß) schreibt die Pakete schließlich auf ein Magnetband.

Die Prozesse: EA, SER, EB und Tape laufen parallel zueinander ab. Die Reihenfolge des zeitlichen Ablaufs muß aber geordnet werden, damit die Datenerfassung korrekt arbeitet. Um dies zu erreichen, müssen die Prozesse synchronisiert werden. Dazu müssen sie miteinander **kommunizieren!**

Während bei /ZWOL 94/ das Problem der Synchronisation klassisch durch Semaphore gelöst wird, wird hier das modernere Prinzip der kommunizierenden sequentiellen Prozesse (CSP) angewendet /HOAR 78/.

4.5. Die Programmiersprache occam 2

Occam 2 ist eine Programmiersprache, die das Prinzip der parallelen CSP für den Benutzer besonders transparent umsetzt. Dies sollte bereits durch den Namen "occam" ausgedrückt werden. Das Grundprinzip des occam-Paten, des im 14. Jh. lebenden englischen Philosophen William of Occam, lautete: "Enita non sunt multiplicandae praeter necessitatem" zu deutsch etwa: "Dinge sollen nicht komplizierter als unbedingt notwendig gemacht werden" /SCHU 88/.

Ein Systementwickler für eine Prozeßdatenverarbeitung sollte in der Lage sein ein paralleles Modell von CSP zu schaffen.
Die Abbildung des Modells mit Hilfe einer sequentiellen Programmiersprache ist aber meist schwierig. Damit wird die Programmierung des Modells komplizierter als die Modellbildung!
Die Struktur von occam 2 erlaubt hingegen ein Modell von CSP direkt durch einfache Befehle zu kodieren. Dabei wird in occam selbst ein einfacher Befehl als eigenständiger Prozeß interpretiert.

Somit liegt der Schwerpunkt der Problemlösung wieder in der Modellbildung.

Das Konzept von occam 2 hat auch Vorteile gegenüber der automatischen Parallelisierung von Programmen, etwa durch ein Betriebssystem, da ein Automat nicht zu einer komplexen parallelen Modellbildung fähig ist.
Sobald ein Problem sich als Modell parallel ablaufender CSP beschreiben läßt, ist occam anderen parallelen Programmiersprachen wie: Algol 68, PL/1, parallel Pascal überlegen /PLES 86/.

Diese Programmiersprache wird daher in Kapitel 5 als Problembeschreibungssprache benutzt. Referenzen hierzu sind: /BAST 91/, /COLO 91/, /ECKE 87/, /EDWA 91/, /FREN 86/, /GEE 91/, /HOAR 78/, /HOAR 85/, /HULS 91/, /KALS 88/, /LAWS 92/, /MANO 85/, /PLES 86/, /SCHU 88/ und /VILL 91/.

4.6. Zusammenfassung

Die Topologie der Architektur der Datenerfassungsanlage für das Experiment COSY 11 ist ein Baum mit zwei Zweigen für die unterschiedlichen Digitalisierungssysteme. Diese realisierte Systemstruktur läßt sich in ein Datenflußdiagramm abbilden. Die so implizierten Prozesse lassen sich als eine Menge von parallel ablaufenden und kommunizierenden sequentiellen Prozessen beschreiben (CSP) /HOAR 78/. Die Programmiersprache occam 2 ist nach diesem Prinzip entwickelt worden und bietet so eine hervorragende syntaktische Unterstützung zur Modellbildung für solche Multiprozessorsysteme. Diese Sprache wird daher im nächsten Kapitel zur Problembeschreibung eingesetzt.

5. Formulierung des Datenerfassungsproblems in occam 2

5.1. Einführung

Diese Einführung erläutert noch einmal die Funktion des Datenerfassungssystems, diesmal mehr aus dem Blickwinkel der ablaufenden Prozesse gesehen.
Das Bild 4-1 zeigt den Aufbau der Hardware für ein Experiment am COSY-Jülich am Beispiel für COSY 11.
Die Analog-Digitalumsetzer in der Digitalisierungsebene empfangen ein Startsignal (Trigger), das den Beginn der Umsetzung einleitet. Einige Zeit danach stehen die Daten eines Meßzyklusses zur Abholung, durch einen Controller oder eine übergeordnete CPU, bereit. Ein Signal (Interrupt) löst dann entsprechende Aktionen der Software aus.
Dabei wird eine Menge von Modulen, die eine logische oder physikalische Einheit bilden, Instrumentierungssystem genannt /ZWOL 94/. In einem Überrahmen-Controller existiert nun für jedes lokal vorhandene Instrumentierungssystem ein Prozeß, der nach einem Startsignal (Trigger) ein Datenpaket generiert.
Dieses Datenpaket erster Ordnung, Subevent genannt, wird dann über den VICbus zu einem SER (Subeventreader), der sich in einem VME-Überrahmen befindet, transportiert. Hier werden die Meßdaten mit Schlüsselinformationen (Header) versehen, um dann wiederum als ein Paket zweiter Ordnung (Teil-Event) über den VMEbus zum EB (Eventbuilder) geleitet zu werden.
Ein Überrahmen mit n Instrumentierungssystemen kann also maximal n verschiedene Subevent-Typen erzeugen.
Der Eventbuilderprozeß läuft in einer schnellen VME-CPU ab. Dieses Programm sammelt und ordnet (synchronisiert) die Subevents und stellt das so entstehende Datenpaket dritter Ordnung (Event) in einem Pufferspeicher zur Verfügung, so daß ein weiterer Prozeß das Sichern dieser Daten auf ein Magnetband übernehmen kann.
Die Realisation dieses Systems ist in /ZWOL 94/ mit Hilfe der prozeduralen Programmiersprache C unter dem Betriebssystem OS-9 durchgeführt worden.
Für die Modellierung des Systems nach Bild 4-1 wird hier occam 2 verwendet, weil sich die Struktur der Datenerfassung einfacher und damit effizienter darstellen läßt.

5.2. Allgemeine Darstellung der Prozesse zur Datenerfassung

5.2.1. Datenflußstruktur

In Bild 5-1 ist zu sehen, wie sich der in Kapitel 4 beschriebene Vorgang der Datenerfassung verallgemeinert darstellen läßt. Hier ist der Datenfluß ohne den Kontrollfluß dargestellt.
Dabei stellt ein Rechteck einen Prozeß (Task) dar, der auf einem Knoten (Node) abläuft. Ein Rechteck mit gestricheltem Rahmen steht für eine Simulation eines Bus-Arbitrierungsprozesses (Bus-Zuteilungsprozesses). Ein Kreis symbolisiert einen Sub-Prozeß (Task oder Thread (Ausführungsfaden)). Die gerichteten Kanten des Graphen sind im erweiterten Sinne der Programmiersprache occam 2 Kanäle, über die Daten transportiert werden können. Eventuelle Pufferspeicher, wie z. B. FIFOs (First in First outs), sind ebenfalls Bestandteil einer Kante /MILD 85/. Dies entspricht nicht /PLES 86/, wo occam-Kanäle als ungepuffert betrachtet werden.

5.2.1.1. Erläuterungen

Mit EA (Event-Acquisition) sind dabei die Prozesse bezeichnet, die nach dem Empfang eines Startsignals (Trigger) Meßdaten in einem Puffer bereitstellen.
Um für einen VICbus-Zweig mehrere EA-Prozesse einfach darstellen zu können, sind die Prozesse in einem Zweig symbolisch, als zweidimensionales Feld indiziert, dargestellt.
D. h. der Prozeß EA{2}{5} liefert Daten über den VICbus-Zweig Nr. 2 und läuft im Controller Nr. 5 ab.
Dies ist sinnvoll, weil man in occam 2 die Möglichkeit hat, mit Hilfe von FOR-Schleifen, mehrere Prozesse zu erzeugen.
Ein Prozeß EA{i}{j} wird dabei über einen Startsignalkanal (Interrupt Request Channel) mit dem Namen c.intreq.ea[i][j] angestoßen.
c.intreq.ea[i][j] heißt: Es handelt sich um einen Kanal c der vom Startsignal intreq zum EA-Prozeß führt. Dieser Kanal ist durch [i][j] in einem Feld von Kanälen indiziert.
Feldkomponenten, die sich nur in dieser Abstraktion abbilden lassen, sind mit geschweiften Klammern { } indiziert, und Feldkomponenten, die sich direkt in occam 2 deklarieren lassen, sind mit eckigen Klammern [] bezeichnet.
Ausgehend von den EA{i}{j} braucht man natürlich Kanäle c.ea.vic[i][j] die die Daten vom Puffer des Controllers zu einem VICbus-Arbitrierungsprozeß: VIC{i} führen. Dies ist in Bild 5-1 nachvollziehbar.
In jedem VIC{i}-Prozeß wird nur ein Datenpaket zu einer Zeit an den Subevent-reader weitergereicht. Es wird also für den VICbus eine Selektion bei gegenseitigem Ausschluß durchgeführt.

Von jedem VIC{i}-Prozeß gelangt man über einen c.vic.ser[i] Kanal zu einem Subeventreader-Prozeß: SER{i}.
Der SER{i}-Prozeß formatiert die Daten zu einem Paket zweiter Ordnung (Teil-Event) und reicht sie über seinen Kanal c.ser.vme[i] an den VMEbus weiter. Der VME-Prozeß selektiert, analog zu VIC{i}, einen Teil-Event für die Ausgabe an seinen Kanal c.vme.eb.
Ein Paket zweiter Ordnung (Teil-Event) ist ein Datenpaket, das von Controllern an einem VICbus-Strang stammt, und welches zur Vervollständigung noch weitere Teil-Events benötigt.
Der Eventbuilder-Prozeß (EB-Prozeß) liest die Daten aus dem Kanal c.vme.eb. Der EB-Prozeß sammelt die Teil-Events und überprüft deren zeitliche Markierungen. Hat er zu einer Markierung genau ns (Anzahl der VICbus-Stränge) Teil-Events gefunden, so ist ein Event komplett, und dieser wird dann über den Kanal c.eb.format an den Formatierungsprozeß übergeben werden.
Der Prozeß FORMAT führt die für die Datenaufzeichnung auf Magnetband notwendige Formatierung durch. Dies geschieht durch Sammeln von Events in sogenannte physikalische Datenblöcke (Records) fester Länge nach dem in der Hochenergiephysik verwendeten ZEBRA-Format /ZOLL 91/ (Kapitel 8.3).
Wenn ein physikalischer Datenblock fertig formatiert ist, wird er über den Kanal c.format.tape zum Gerätetreiber für das Magnetband transportiert.
Damit ist die Aufzeichnung eines oder einiger Events abgeschlossen.

5.2.2. Kontroll- und Datenfluß

In Bild 5-2 sind auch die notwendigen Kanäle für den Kontrollfluß dargestellt. Die Indizierung erfolgt analog zu Kapitel 5.2.1.

5.2.2.1. Erläuterungen

Im Unterschied zu Bild 5-1 sind hier auch Kanäle entgegen der Datenflußrichtung eingezeichnet. Diese sind für die Steuerung der Prozesse unerläßlich.
In /ZWOL 94/ werden für die Synchronisation Semaphore verwendet. Diese Funktion wird hier, bei Prozessen (Tasks) in verschiedenen Zentraleinheiten generell, durch den Rendevous-Mechanismus der Kanäle übernommen. Befinden sich aber zwei Ausführungsfäden (Threads) in der gleichen Zentraleinheit, dann ist die Verwendung von Semaphoren meist vorteilhafter /3L 93.8/.

Jeder der Prozesse: EA, SER, EB, FORMAT und TAPE (nicht dargestellt) besitzt einen Steuerkanal: c.vic.ea[i][j], c.vme.ser[i], c.ui.eb, c.eb.format und c.format.tape. Beim Start der Datenerfassung sendet der Experimentator mit Hilfe des Benutzerschnittstellenprozesses (UI = User Interface) ein Startsignal zum Eventbuilder. Dieser gibt eine Startnachricht über seinen Steuerkanal c.eb.vme

weiter. Das Startkommando wird dann über die Baumstruktur der Steuerkanäle bis zu den EA-Prozessen verteilt. Dadurch wird die Datenerfassung gestartet.
Ein SER wartet dann auf eine Nachricht auf seinem c.vic.ser[i]-Kanal, die anzeigt, daß er nun Daten auslesen kann. Gleichzeitig sammeln die EA-Prozesse die in den Instrumentierungssystemen anfallenden Daten. Analog wartet der EB auf eine Nachricht von einem der SER, FORMAT wartet auf den EB und Tape wartet auf FORMAT.
Für diese Struktur, die ein Modell von /ZWOL 94/ bildet, ist festzuhalten, daß immer nur ein SER{i} mit dem EB gleichzeitig kommunizieren kann.

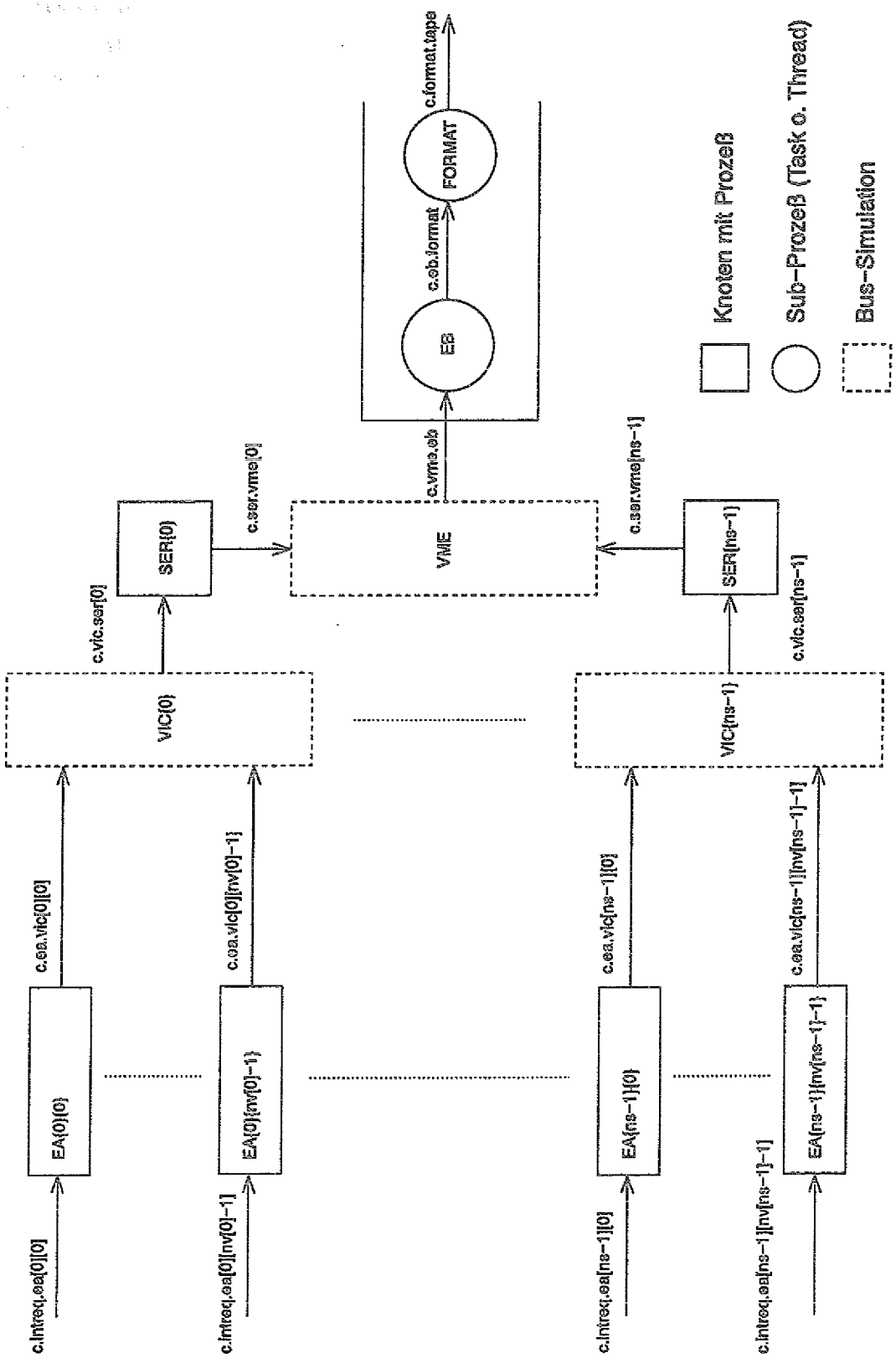


Bild 5-1: Die Datenflußstruktur der Datenerfassungssysteme für COSY-Jülich

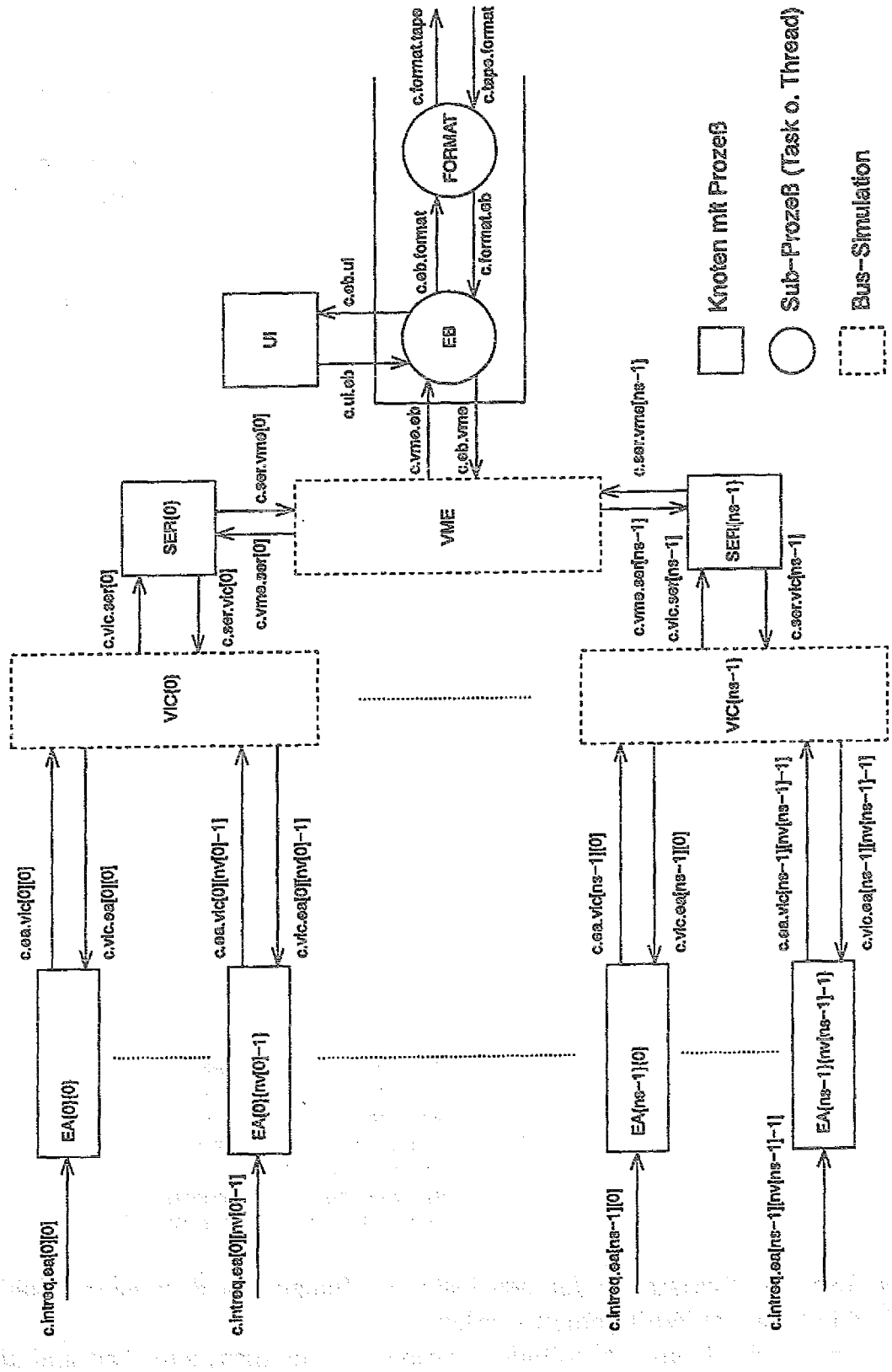


Bild 5–2: Die Kontroll– und Datenflußstruktur der Datenerfassungssysteme für COSY–Jülich

5.2.2.2. Erläuterungen der Indizes

Die Indizes in den jeweils unteren Zweigen sind zunächst etwas unverständlich und sollen hier näher erläutert werden.
In Bild 5-2 links oben ist der 0-te VICbus-Zweig skizziert. An diesem Zweig sind eine Anzahl nv[0] von Controllern angeschlossen, die von 0 bis nv[0]-1 durchnumeriert sind. nv[i] ist die dem VICbus-Zweig Nr. i zugeordnete Anzahl von Controllern und das Element eines eindimensionalen Feldes.
Links unten ist der ns-1 -te VICbus-Zweig dargestellt. Es gibt ns Zweige mit den Indizes 0 bis ns-1. An den ns-1 -ten VICbus-Zweig sind nv[ns-1] Controller angeschlossen. Somit hat also der am höchsten zu indizierende Prozeß den Index {ns-1}{nv[ns-1]-1}.
Die Indizes der angeschlossenen Kanäle sind identisch.

5.3. Abbildung der Struktur in ein occam 2 Programm

5.3.1. Globale Deklarationen

Die folgenden Deklarationen gelten für die in Kapitel 5.3 und 5.4 gezeigten Programmfragmente:

```
VAL a.nv IS [3,6,4,5]:      -- Angabe über die konkrete
                            -- Experimentkonfiguration: Anzahl
                            -- der EA am i-ten VICbus-Zweig
VAL ns IS 4:                -- !!weil oben vier Werte stehen
VAL nv.max IS 6:            -- max. Anzahl von EAs
                            -- an einem VICbus-Zweig
                            -- !!siehe weiter oben
VAL buf.siz IS 2048:        -- Größe sämtlicher Puffer

[ns][nv.max] INT a.ni:     -- Anzahl Instrumentierungssysteme:
                           -- i-ter VICbus j-ter Überrahmen
[ns][nv.max] CHAN OF INT c.intreq.ea:
[ns][nv.max] CHAN OF INT::[]INT c.ea.vic, c.vic.ea:
[ns]         CHAN OF INT::[]INT c.vic.ser, c.ser.vic,
                                c.ser.vme, c.vme.ser:
             CHAN OF INT::[]INT c.vme.eb, c.eb.vme,
                                c.ui.eb, c.eb.ui,
                                c.eb.format, c.format.eb,
                                c.format.tape, c.tape.format:
```

Dabei ist INT die Deklaration für den Datentyp Integer, in dem ganze Zahlen durch ein 4 byte langes Wort kodiert werden.
Die Konstanten oder Konstanten-Felder (-arrays): ns, nv.max, a.nv und a.ni sind für ein hypothetisches System festgesetzt, welches 4 VICbus-Zweige mit jeweils 3, 6, 4 bzw. 5 angeschlossenen Controllern besitzt.

Die Kanalprotokolle von occam 2 gestatten es in eleganter Weise, die für das System nach Bild 5-2 benötigten Kanäle zu deklarieren. Dabei ist z. B. [ns][nv.max] CHAN OF INT::[]INT c.ea.vic: die Deklaration eines zweidimensionalen Feldes von Kanälen, über die Integer-Datenpakete variabler Länge geschickt werden können. Die Angabe der aktuellen Länge erfolgt im Integer-Format zu Beginn des Datenpakets /SCHU 88/. Eine ähnliche Datenstruktur wird auch für die COSY-Datenerfassung benutzt.

5.3.2. Abbildung der EA{i}{j}-Prozesse in occam 2

Das folgende Programmfragment ist die Beschreibung der Event-Acquisitions-prozeduren p.event.acq(i, j) (Datenquellen) in occam 2:

```
-- EA-Prozesse
PAR i = 1 FOR ns
  PAR j = 1 FOR a.nv[i] -- (ns * a.nv[i]) EA-Prozesse
    p.event.acq(i, j)   -- Prozeduraufruf!
                        -- => Generation von Prozessen!
```

Durch diese Zeilen werden ns Gruppen von Prozessen erzeugt. Der i-ten Gruppe gehören dabei genau a.nv[i] EA-Prozesse an. Hinter der Prozedur p.event.acq(i, j) verbirgt sich jedoch eine Schar von unterschiedlichen Routinen, die durch Übergabe der Parameter i und j selektiert werden. Diese Routinen sind anwendungs- und systembedingt verschieden, da bei COSY mit unterschiedlich bestückten Digitalisierungssystemen gearbeitet wird.
Die Struktur der EA-Prozesse ist in Kapitel 5.4.1 erläutert.

5.3.3. Abbildung der VIC{i}-Prozesse in occam 2

Das folgende Programmfragment ist die Beschreibung der VMEbus-Arbitrierungsprozesse: VIC{i}:

```
-- VIC-Prozesse
PAR i = 1 FOR ns -- ns parallele VIC-Prozesse
  INT isize, osize, message:  -- lokaler Puffer
  [buf.siz] INT a.idata, a.odata:
  WHILE TRUE
    ALT
      c.ser.vic[i] ? 1::message  -- Weitergabe einer Kontroll-
                                 -- Nachricht
        c.vic.ea[i][message>>16] ! message
      ALT j = 1 FOR a.nv[i] -- a.nv[i] alternative
                            -- Eingabe-Prozesse
        c.ea.vic[i][j] ? isize::a.idata -- Guard
          SEQ
            a.odata := a.idata   -- Einlesen und weiterreichen
            osize := isize
            c.vic.ser[i] ! osize::a.odata
```

In diesem Fragment werden weitere mächtige Sprachelemente von occam 2 benutzt. Das Kürzel: SEQ (SEQuenzkonstruktor) markiert, daß die folgenden Prozesse sequentiell abgearbeitet werden sollen. Die Zeile: "PAR i = 1 FOR ns" (PAR = PARallelkonstruktor) erzeugt genau ns parallel ablaufende Prozesse, die durch den Index i unterschieden werden. Für den i-ten Subeventreader wird so ein Prozeß: VIC{i} erzeugt.
Der Befehl "ALT j = 1 FOR a.nv[i]" (ALT = ALTernativkonstruktor) führt zur Erzeugung von a.nv[i] Prozessen, von denen bei jedem Schleifendurchlauf genau einer ausgeführt wird, falls gerade keine Steuerungsnachricht übertragen werden muß. Dabei reagiert ein Eingabeprozeß auf den zuerst aktivierten Controller über einen Kanal c.ea.vic[i][j] /SCHU 88, S.138/ und der Ausgabeprozeß gibt das eingelesene Datenpaket auf den Ausgabekanal c.vic.ser[i] aus. Im Vokabular von occam 2 nennt man den Eingabeprozeß nach dem ALT-Konstruktor (Hier: "c.ea.vic[i][j] ? isize::a.idata") einen Guard (Wächter). Dieser "wacht" darüber, welcher der möglichen Prozesse ausgeführt werden darf.
Die Kontrollnachrichten sollten möglichst kurz sein. Sie bestehen hier aus 4 byte Wörtern (Integer-Format). Die höherwertigen 2 byte enthalten die Überrahmennummer, die niederwertigen 2 byte enthalten die Information.

5.3.3.1. Erläuterungen

Ein VICbus-Prozeß simuliert einen VICbus. Er überträgt einen Subevent von einem EA-Prozeß zu seinem SER oder ein Steuerkommando von diesem SER zu einem der EA-Prozesse und damit zu den Controllern.

Die Zeile: "c.intreq.ea[i][j] ? dummy" ist die occam 2 Notation für einen Eingabeprozeß aus dem Kanal c.intreq.ea[i][j]. Das Kanalprotokoll ist dabei als CHAN OF INT deklariert, d. h. über diesen Kanal wird im aktiven Zustand ein Integer-Wert transportiert. Dies bewirkt die Synchronisation auf das externe Startsignal, ohne daß das übertragene Datum ausgewertet werden muß.
Mit dem Ausgabeprozeß: "c.ea.vic[i][j] ! osize::a.odata" werden die erfaßten Daten zum Kanal c.ea.vic[i][j] ausgegeben.
Man sieht, wie effizient sich die Prozesse: EA{i}{j} in occam 2 abbilden lassen.

5.3.4. Abbildung der SER{i}-Prozesse in occam 2

Das folgende Programmfragment ist die Beschreibung der Subeventreader-Prozesse: SER{i} in occam 2:

```
--SER-Prozesse
PAR i = 1 FOR ns -- ns parallele SER-Prozesse
  p.subevent.read(i)  -- Prozeduraufruf!
                                -- => Generation von Prozessen!
```

Es wird wieder ein replikatives PAR benutzt, um die ns Subeventreader-Prozeduren hinzuschreiben. Die Subeventreader lesen ihre Daten aus dem jeweiligen Kanal c.vic.ser[i] und bearbeiten sie mit ihrem Prozeß, der durch die Prozedur p.subevent.read(i) definiert wird.
Nähere Erläuterungen sind in Kapitel 5.4.2 gegeben.

5.3.5. Abbildung des VME-Prozesses in occam 2

Das folgende Programmfragment ist die Beschreibung des VMEbus Arbitrierungsprozesses in occam 2:

```
-- VME-Prozeß
INT isize, osize, message:     -- lokaler Puffer
[buf.siz] INT a.idata, a.odata:
WHILE TRUE
  ALT
    c.eb.vme ? 1::message  -- Weitergabe einer Kontroll-
                           --                  Nachricht
      c.vme.ser[message>>16] ! message
    ALT i = 1 FOR ns -- ns alternative Eingabe-Prozesse
      c.ser.vme[i] ? isize::a.idata -- Guard
        SEQ
          a.odata := a.idata  -- Einlesen und weiterreichen
          osize := isize
          c.vme.eb ! osize::a.odata
```

Dieser Prozeß funktioniert analog zu 5.3.3, es gibt hier aber nur ein Bündel von alternativen Prozessen, da auch nur ein VMEbus in diesem System vorhanden ist.

5.3.6. Abbildung des EB-Prozesses in occam 2

Das folgende Programmfragment ist die Beschreibung der EB (Eventbuilder) Prozedur in occam 2:

```
-- EB-Prozeß
p.event.build()  -- Prozeduraufruf => Prozeßgeneration!
```

Die Prozedur p.event.build führt so komplexe Funktionen aus, daß hier nur der Prozeduraufruf im Hauptprogramm gezeigt wird. Eine nähere Erklärung des Prozesses ist in Kapitel 5.4.3 gegeben.

5.3.7. Abbildung des FORMAT-Prozesses in occam 2

Das folgende Programmfragment ist die Beschreibung der FORMAT-Prozedur in occam 2:

```
-- FORMAT-Prozeß
p.format()  -- Prozeduraufruf => Prozeßgeneration!
```

Der Prozeß FORMAT führt die für die Datenaufzeichnung auf Magnetband notwendige Formatierung durch. Dies geschieht durch Sammeln von Events in sogenannte physikalische Datenblöcke (Records) fester Länge nach dem in der Hochenergiephysik verwendeten ZEBRA-Format /ZOLL 91/ (Kapitel 8.3).

5.3.8. Das Hauptprogramm

Das occam 2 Hauptprogramm, das sich aus Bild 5-2 ergibt, findet man leicht durch Auflisten der bisher erläuterten Prozeduren. Sie sind nur noch durch einen Parallelkonstruktor (PAR) einzuklammern, so daß die verschiedenen Stufen des Datenerfassungssystems parallel ablaufen können. Hier das Hauptprogrammfragment, das die Struktur von Bild 5-2 in occam 2 abbildet:

```
-- COSY-Datenerfassung
PAR
  -- FORMAT-Prozeß
  p.format()  -- Prozeduraufruf!
  -- EB-Prozeß
  p.event.build()  -- Prozeduraufruf!
  -- VME-Prozeß
  INT isize, osize, message:  -- lokaler Puffer
  [buf.siz] INT a.idata, a.odata:
  WHILE TRUE
    ALT
      c.eb.vme ? 1::message -- Weitergabe einer Kontroll-
                            --                Nachricht
        c.vme.ser[message>>16] ! message
      ALT i = 1 FOR ns -- ns alternative Eingabeprozesse
        c.ser.vme[i] ? isize::a.idata -- Guard
          SEQ
            a.odata := a.idata   -- Einlesen und weiterreichen
            osize := isize
            c.vme.eb ! osize::a.odata
  -- SER-Prozesse
  PAR i = 1 FOR ns -- ns parallele SER-Prozesse
    p.subevent.read(i)  -- Prozeduraufruf!
  -- VIC-Prozesse
  PAR i = 1 FOR ns -- ns parallele VIC-Prozesse
    INT isize, osize, message:   -- lokaler Puffer
    [buf.siz] INT a.idata, a.odata:
    WHILE TRUE
      ALT
        c.ser.vic[i] ? 1::message  -- Weitergabe einer
                                   -- Kontroll-Nachricht
          c.vic.ea[i][message>>16] ! message
        ALT j = 1 FOR a.nv[i] -- a.nv[i] alternative
                              -- Eingabeprozesse
          c.ea.vic[i][j] ? isize::a.idata -- Guard
            SEQ
              a.odata := a.idata   -- Einlesen und weiterreichen
              osize := isize
              c.vic.ser[i] ! osize::a.odata
  -- EA-Prozesse
  PAR i = 1 FOR ns
    PAR j = 1 FOR a.nv[i] -- (ns * a.nv[i]) EA-Prozesse
      p.event.acq(i, j) -- Prozeduraufruf!
```

5.4. Die Prozesse des occam 2-Programms zur Datenerfassung

5.4.1. Die Event-Acquisitions-Prozesse

Ein solcher Prozeß läuft in der CPU eines intelligenten Controllers, dessen Überrahmen mit Analog-Digital- oder Zeit-Digital-Umsetzern bestückt ist. Dabei ist hier vorausgesetzt, daß beim Laden des Programms Zusatzinformationen, wie z. B. die Überrahmen-Nummer und eine Tabelle mit Bestückungsinformationen in Form einer Modulliste, übergeben werden. Die interne Struktur der Prozesse: EA{i}{j} ist in Bild 5-3 dargestellt.

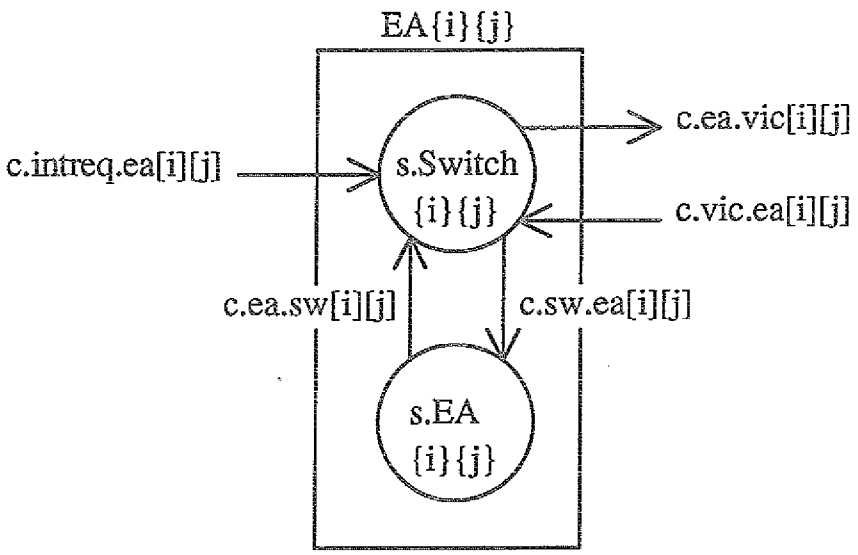


Bild 5-3 Die interne Struktur der Prozesse: EA{i}{j}

Ein Prozeß EA{i}{j} gliedert sich in zwei Sub-Prozesse: s.Switch{i}{j} (Schalter) und s.EA{i}{j}. Dabei bearbeitet der s.Switch-Prozeß ausschließlich die Kommunikationsanforderungen, und s.EA führt parallel (!) dazu das Aufsammeln der Daten durch. Die Indizes: {i}{j} sind im folgenden stellenweise weggelassen, damit der Text besser lesbar wird.
Daten von s.EA werden zu s.Switch nicht Datum für Datum (per value), sondern einfach durch die Übergabe eines Zeigers (per reference) übertragen. Dies ist möglich, weil der Transfer in **einer** Zentraleinheit mit **einem** gemeinsamen Speicherbereich stattfindet.

Der Prozeß s.Switch läßt sich leicht in occam 2 abbilden. Hier eine fragmentarische Version des Kodes:

```
[ns][nv.max] CHAN OF INT::[]INT c.ea.sw, c.sw.ea: -- Deklaration
                                  -- der Kanäle zum Schaltprozeß
[msg.max]    INT a.message:
             INT msg.size:

ALT
  c.intreq[i][j] ? msg.size::a.message -- Guard für einen
                                       -- Interrupt
    --
    -- Interrupt-Behandlung
    --
  c.vic.ea[i][j] ? msg.size::a.message -- Guard für ein
                                       -- Steuerkommando
    --
    -- Steuerkommando bearbeiten
    --
  c.ea.sw[i][j] ? msg.size::a.message  -- Guard für eine
                                       -- Statusmeldung
    --
    -- Statusmeldung bearbeiten
    --
```

Die Funktion des Prozesses s.EA läßt sich besser durch ein Struktogramm erklären (Bild 5-4).

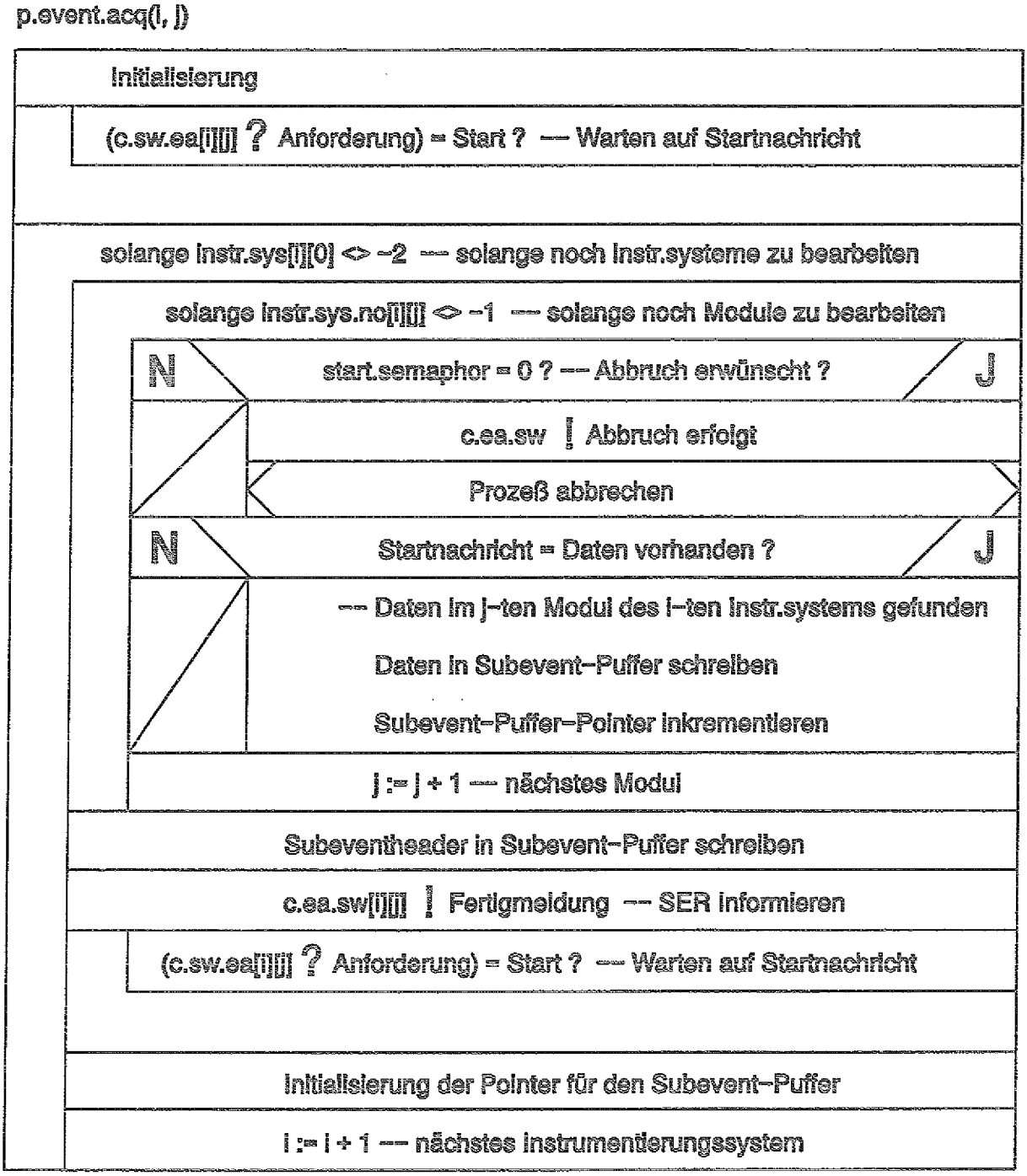


Bild 5-4: Struktogramm einer Prozedur p.event.acq(i, j)

5.4.1.1. Erläuterungen

Der Prozeß wird vom s.Switch-Prozeß aktiviert, und startet nach Erhalt einer Nachricht über den Kanal c.sw.ea[i][j].
Das Programm wird durch die Liste a.instr.sys[i][j] gesteuert. In dieser Liste steht für das j-te Modul des i-ten Instrumentierungssystems die physikalische Platznummer (Slot) im Überrahmen. Der Wert -2 in a.instr.sys[i][0] markiert das Listenende. Der Wert -1 zeigt an, daß in diesem Instrumentierungssystem keine weiteren Module vorhanden sind. So können, durch zwei ineinander verschachtelte Schleifen, alle Module abgefragt werden.
Aus der Liste a.config[i][j] wird entnommen, wieviele Daten aus dem entsprechenden Modul auszulesen sind.
Erst wenn alle Module eines Instrumentierungssystems abgefragt sind, kann der Subevent-Dateischlüssel (Header) aufgesetzt werden. Damit ist der Subevent fertig und er wird, durch Übergabe eines Zeigers, an den Schaltprozeß übergeben und von ihm auf den Ausgabekanal c.ea.vic[i][j] übertragen.
Nach der erneuten Initialisierung der Subevent-Puffer-Zeiger (Pointer) und der Inkrementierung der Schleifenvariablen wird das Programm mit der Betrachtung des nächsten Moduls fortfahren. Dies geschieht aber erst, wenn der SER diesen Subevent geholt hat.
Es ist wichtig, daß ein Instrumentierungssystem auch dann einen Subevent erzeugt, wenn keines der Module Daten bereithält. Nur so kann der Subeventreader nachzählen, ob er alle Subevents von den angeschlossenen Überrahmen empfangen hat.
Der Prozeß s.EA kann außerdem durch Steuerung über eine Semaphore abgebrochen werden.

5.4.2. Der Subeventreader-Prozeß

Wie in Kapitel 5.3 beschrieben, werden die Subevents einem VIC{i}-Prozeß zugeführt. Dieser Prozeß ist ein Modell für die auf dem VICbus ablaufende Arbitrierung.
Am Ausgang von VIC{i} erscheinen dann die Subevents in ungeordneter Reihenfolge wieder. Der Subeventreader liest diese Daten und stellt sie in einem geordneten Paket zusammen.

Um die Prozedur besser verstehen zu können, wird hier das Dateiformat eines Subevents vorgestellt:

Wort 1	Subevent Länge: NSEWL		\	
Wort 2	Crate-Nr: CN	Instr.-Sys.-Nr.	\	Subevent-Dateischlüssel
Wort 3	Oper.-Mode	Sub-Type: ST	/	
Wort 4	Start-Bus Zählerinformation		/	
Wort 5	Moduladr.	Wortanzahl		Modul-Datenschlüssel
Wort 6	Kanal-Nr.	Daten		
Wort 7	Kanal-Nr.	Daten		Modul 1
Wort 8	Kanal-Nr.	Daten		
Wort	.....	.....		
Wort ...	Moduladr.	Wortanzahl		Modul-Datenschlüssel
Wort	Kanal-Nr.	Daten		
Wort	Kanal-Nr.	Daten		Modul n
Wort NSEWL	Kanal-Nr.	Daten		

Die Datenblöcke der Module werden sequentiell aneinandergereiht und jeweils durch die Moduladresse und die Angabe der Wortanzahl der direkt folgenden Daten eingeleitet. Diese Datenmenge wird mit dem Subevent-Dateischlüssel versehen. In ihm steht die gesamte Länge des Subevents, woher die Daten stammen, und wann sie erfaßt wurden. Der Startsignal-Bus (Trigger-Bus) sorgt dafür, daß in allen angeschlossenen Überrahmen die gleiche relative Zeitangabe, in Form eines Zählerstands, vorliegt /ESSE 92/ /ZWOL 94/.

Analog zu Kapitel 5.4.1 haben auch die Prozesse: SER{i} eine innere Struktur.

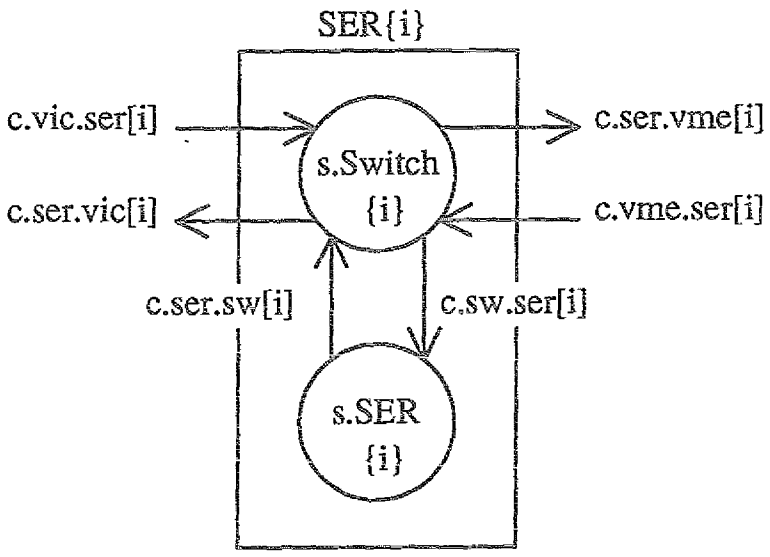


Bild 5-5 Die interne Struktur der Prozesse: SER{i}

Wieder erledigt ein s.Switch-Prozeß (Schaltprozeß) die Interprozeßkommunikation. Der s.SER-Prozeß wird durch Kommandos gesteuert und gibt Statussignale zurück. Die Datenübertragung innerhalb dieser Prozesse erfolgt, per Referenz, durch Zeiger.

Das Struktogramm einer Prozedur p.subevent.read(i) ist in Bild 5-6 wiedergegeben.

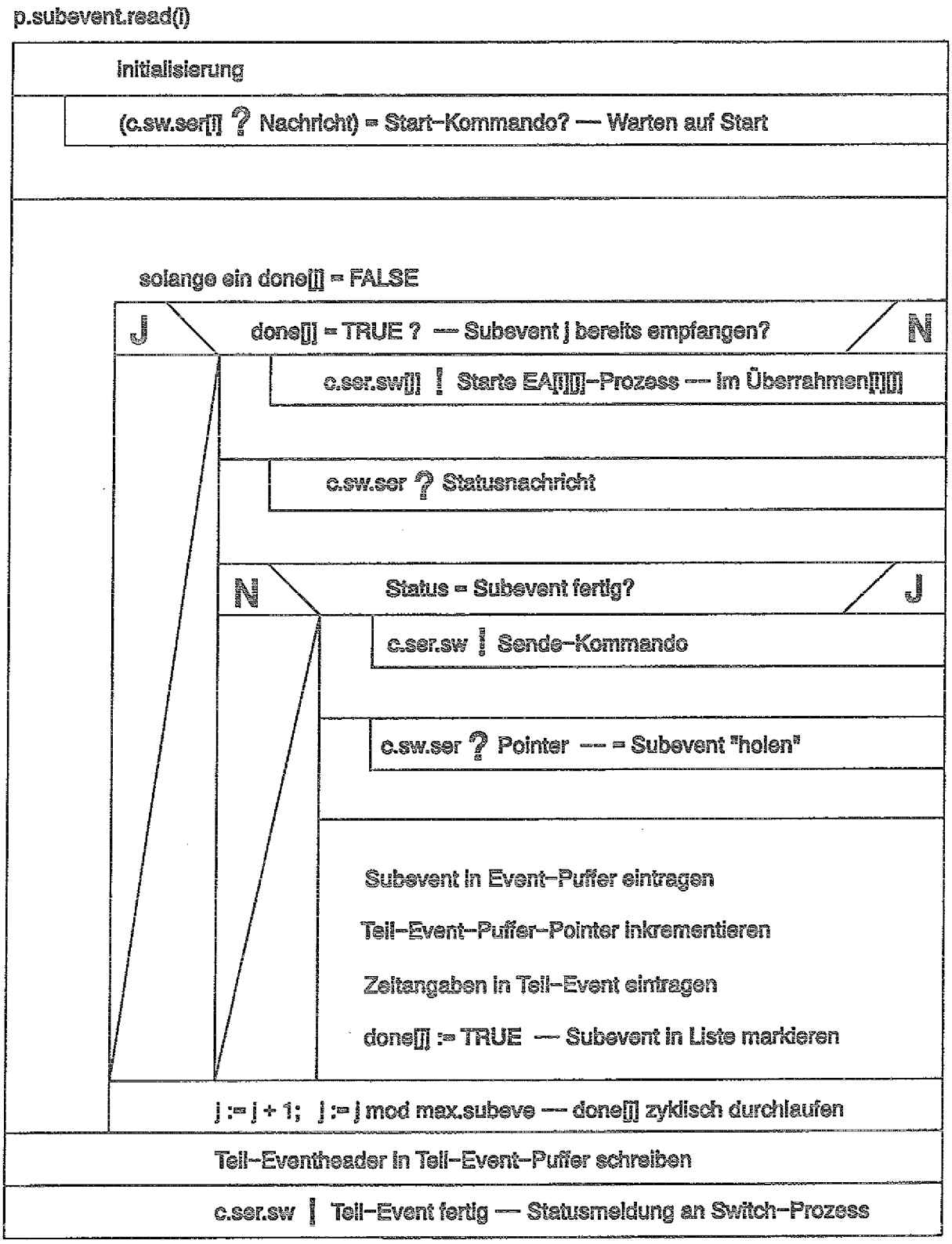
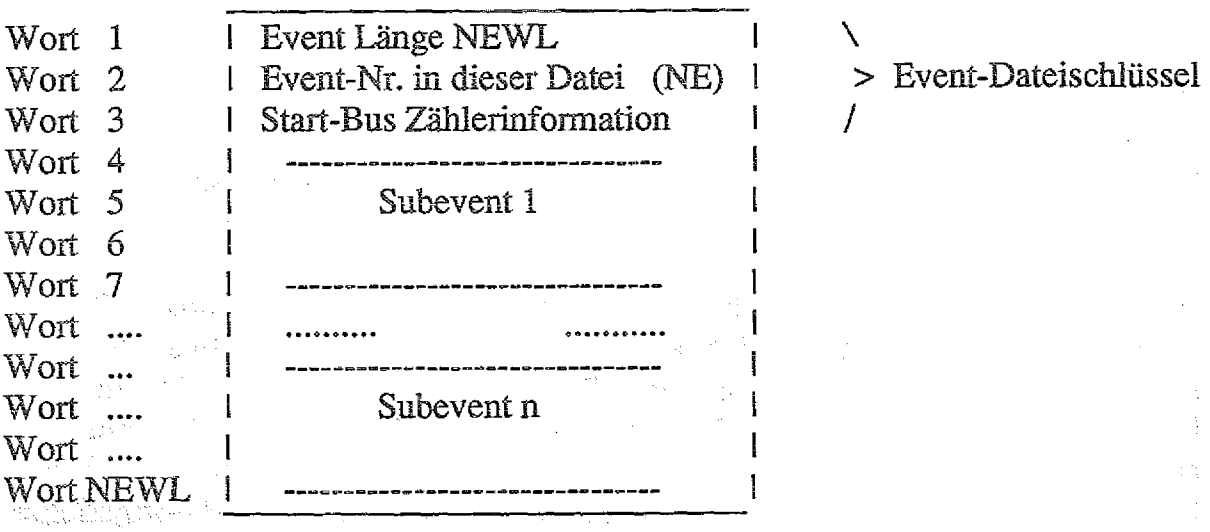


Bild 5-6: Struktogramm einer Prozedur p.subevent.read(i)

5.4.2.1. Erläuterungen

Nach einem Initialisierungsteil wird auf das Eintreffen der Startnachricht vom Schaltprozeß, über den Kanal c.sw.ser[i], gewartet. Wenn der Eventbuilder das Startsignal bereits gesendet hatte, kann der SER (Subeventreader) sofort mit seiner Arbeit beginnen. Der SER läuft durch eine Liste und findet so heraus, welche Subevents er noch braucht, um einen Teil-Event zusammenzustellen. Die Überrahmen, die noch Daten liefern müssen, werden also immer wieder zyklisch angesprochen (Polling-Verfahren).
Der s.subevent.read-Prozeß erhält einen Zeiger auf einen angekommenen Subevent. Der Subevent wird in einen Puffer für den Teil-Event geschrieben und in der Liste markiert (abgehakt).
Ein vollständiger Teil-Event wird mit einer Schlüsselinformation (Header) versehen und zum VMEbus ausgegeben. Dazu wird dem Schaltprozeß ein Zeiger auf den auszugebenden Puffer übergeben. Der Schaltprozeß gibt dann den Teil-Event über einen Kanal c.ser.vme[i] und den Kanal c.vme.eb an den Eventbuilder weiter. Ein Teil-Event ist ein Datenpaket, das nur aus den Subevents der Instrumentierungssysteme an einem VICbus-Strang besteht.

Um die Prozedur besser verstehen zu können, wird hier das Format eines Teil-Events, bzw. kompletten Events, vorgestellt:



Im ersten Datenwort steht die Länge des gesamten Teil-Events. Die Information im Datenwort Nr. 2 (Event-Nr. in dieser Datei) wird erst zum Zeitpunkt der Speicherung auf Magnetband eingetragen, da diese sich erst durch die Formatierung ergibt. An dieser Stelle wird der Platz für Wort Nr. 2 aber schon geschaffen, um mit einheitlichen Dateiformaten für Teil-Events und Events arbeiten zu können. Der Datenabschnitt des Teil-Events besteht dann aus den aneinandergereihten Subevents.

5.4.3. Der Eventbuilder-Prozeß

Dieser Prozeß leistet dasselbe wie unter 5.4.2 beschrieben: Die SER{i} packen aus Subevents Teil-Events, während der Eventbuilder-Prozeß aus Teil-Events komplette Events formt. Der entsprechende Schaltprozeß muß aber noch zusätzlich direkt mit der Benutzerschnittstelle (UI = User-Interface) kommunizieren können, da der Benutzer über den Prozeß UI alle anderen Prozesse direkt oder indirekt steuert (vgl. Bild 5-2). Der Anwender ist also die Quelle aller interaktiven Steuerkommandos.
Aus Teil-Events wird nach dem Entfernen der Dateischlüssel (Header) ein Event zusammengestellt. Die Datenquelle ist dabei der VMEbus und die Datensenke ist das Formatierungsprogramm für das Magnetband. Die Anzahl der Teil-Events die für einen Event benötigt werden, ist gleich der Anzahl der vorhandenen Subeventreader: ns. Das setzt voraus, daß nach einen Startimpuls (Trigger-Ereignis) alle Subeventreader einen Teil-Event erzeugen und weiterreichen, auch wenn gar keine Daten zu transportieren sind. So kann der Eventbuilder einfach mitzählen und selbst feststellen, wann ein Event komplett ist.

5.5. Zusammenfassung

Die allgemeine Struktur der Datenerfassungssysteme am COSY-Jülich wird als Graph dargestellt. Diese Struktur läßt sich in ein Konzept von occam 2-Prozeduren abbilden.
Dieses Konzept stellt einen algorithmischen Fahrplan zur Realisierung von Datenerfassungssystemen für COSY-Experimente bereit. Dabei dient occam 2 als formale Programmiersprache zur Problembeschreibung.
Im nächsten Kapitel wird ein Konzept zur Implementierung eines solchen Datenerfassungssystems, am Beispiel des Experiments COSY 11, beschreiben.

Hier ist nicht erläutert, wie die Programme im System ihre Knoten (Nodes) erreichen. Dazu ist es erforderlich, auf dem Experimentrechner Software bereitzustellen, welche die Konfigurierung und Steuerung der Knoten via Ethernet oder Ethernet und VICbus ermöglicht. Dabei sollen Funktionen wie Verbindungsaufbau nach TCP/IP (Transport Control Protocol / Internet Protocol), Herunterladen (Download) und Zurücklesen (Upload) von Konfigurationsinformationen, Herunterladen und Zurücklesen von Programmen, Acquisitionsprogrammstart und -stop, ferngesteuert und mit einer für alle Instrumentierungssysteme einheitlichen Benutzerschnittstelle realisiert werden. Dazu findet sich in /WUES 92/ /ZWOL 92/ /ZWOL 93/ /ZWOL 94/ ein Lösungsansatz, der auf der Anwendung einer Teilmenge der von MAP (Manufacturing Automation Protocol) /GM 88/ her bekannten MMS-Software (Manufacturing Message Specification) /ISO 90/ basiert.

6. Implementierungskonzept für ein optimiertes System

Das in den Kapiteln 4 und 5 beschriebene System ist nicht optimal. Daher wird in Kapitel 6.1 ein Vorschlag für eine bessere Systemstruktur aufgezeigt.
In Kapitel 6.2 wird das Konzept zur Realisierung der Hardware vorgestellt. Kapitel 6.3 befaßt sich mit der Erläuterung der Software-Entwicklungsumgebung. In Kapitel 6.4 werden dann die Komponenten und die komplette Architektur der optimierten Hardware genauer beschrieben.

6.1. Die optimierte Systemstruktur

Die in Bild 5-2 gezeigte Struktur hat Engpässe (Bottlenecks) bezüglich der möglichen Datenübertragungsrate. Die Schwäche dieser Struktur sind die Systembusse, da über sie immer nur ein Kanal **gleichzeitig** zwischen zwei Prozessen geöffnet werden kann (Zeitmultiplex-Verfahren). Übersteigt die Summe der erforderlichen Datentransferrate der am Bus angeschlossenen Teilnehmer die Bandbreite des Busses, dann kann das System seine Aufgaben nicht mehr erfüllen. Eine parallelisierte Struktur, in der bei n Teilnehmern n Kanäle geöffnet werden können, wäre ideal. In der Praxis ist bei Transputern typisch: n = 4, oder beim T9000 Transputer bzw. TMS320C40 Signalprozessor: n = 6.

In Bild 6-1 ist die Struktur gezeigt, die nach der Analyse von Bild 5-2 entworfen worden ist.

Die linke Seite von Bild 5-2 stimmt mit der linken Seite von Bild 6-1 überein, da auf den VICbus, aufgrund der CANU-Empfehlungen, nicht verzichtet werden kann. Das ist aber auch nicht notwendig, da die Bandbreite eines VICbus-Stranges von bis zu 33 Mbyte/s ausreichend ist /ISO 93/ /ZWOL 91/ /ZWOL 92/ /ZWOL 94/.
Bei einem System mit z. B. drei VICbus-Strängen, müßte der VMEbus eine Bandbreite von 3 x 33 Mbyte/s = 99 Mbyte/s haben, um den Datenfluß nicht zu behindern. Der VMEbus hat jedoch - theoretisch - nur eine Bandbreite von 40 Mbyte/s, daher wird er durch direkte Verbindungen ersetzt.
Ein Multiprozessorsystem mit dem VMEbus ist also schlecht skalierbar, weil die Kommunikationsbandbreite zur Interprozessorkommunikation konstant ist.
In einem Transputersystem hingegen, vergrößert sich mit jedem neuen Prozessor auch die Kommunikationsbandbreite, d. h. die Gesamtleistung des Systems ist gut skalierbar.
Um die Bandbreite der Kanäle optimal ausnutzen zu können, ist es besser, die Daten der Subevents zu komprimieren. Dies wird durch den Kompressionsprozeß s.KMP{i} in jedem Subeventreader-Prozeß SER{i} bewirkt. Das System soll aber

auch ohne Kompression funktionieren, die Aktivierung des Kompressionsprozesses ist daher optional.
Ein Beispiel für einen Kompressionsalgorithmus ist eine Schwerpunktsbildung, d. h. hat ein Teilchenstrahl mehrere Detektorelemente zum Ansprechen gebracht, kann die entstehende Datenmenge durch Errechnen des tatsächlichen Durchflugpunkts auf einen Wert reduziert werden. Man kann so Kompressionsraten in der Größenordnung 1 zu 5 erreichen. Damit hat man die Möglichkeit, im Multiprozessorsystem fehlende Kommunikationsbandbreite durch Rechenleistung zu ersetzen. So kann man für ein ausgewogenes Verhältnis dieser beiden Größen sorgen.
Bei einer mittleren Kompressionsrate von 1 zu 5 kann die Datenerfassungsrate um das 5-fache gesteigert werden. Bei der Verwendung eines Magnetbands mit einer Übertragungsrate von 1,25 Mbyte/s ergibt sich ein Wert von 5 x 1,25 Mbyte/s = 6,25 Mbyte/s!
Zwischen Eventbuilder- und Formatierungsprozeß ist ein Testprozeß eingefügt. Dieser überprüft die Events auf ihre physikalische Plausibilität, so werden physikalisch unmögliche Events herausgefiltert und damit die Datenrate weiter reduziert.

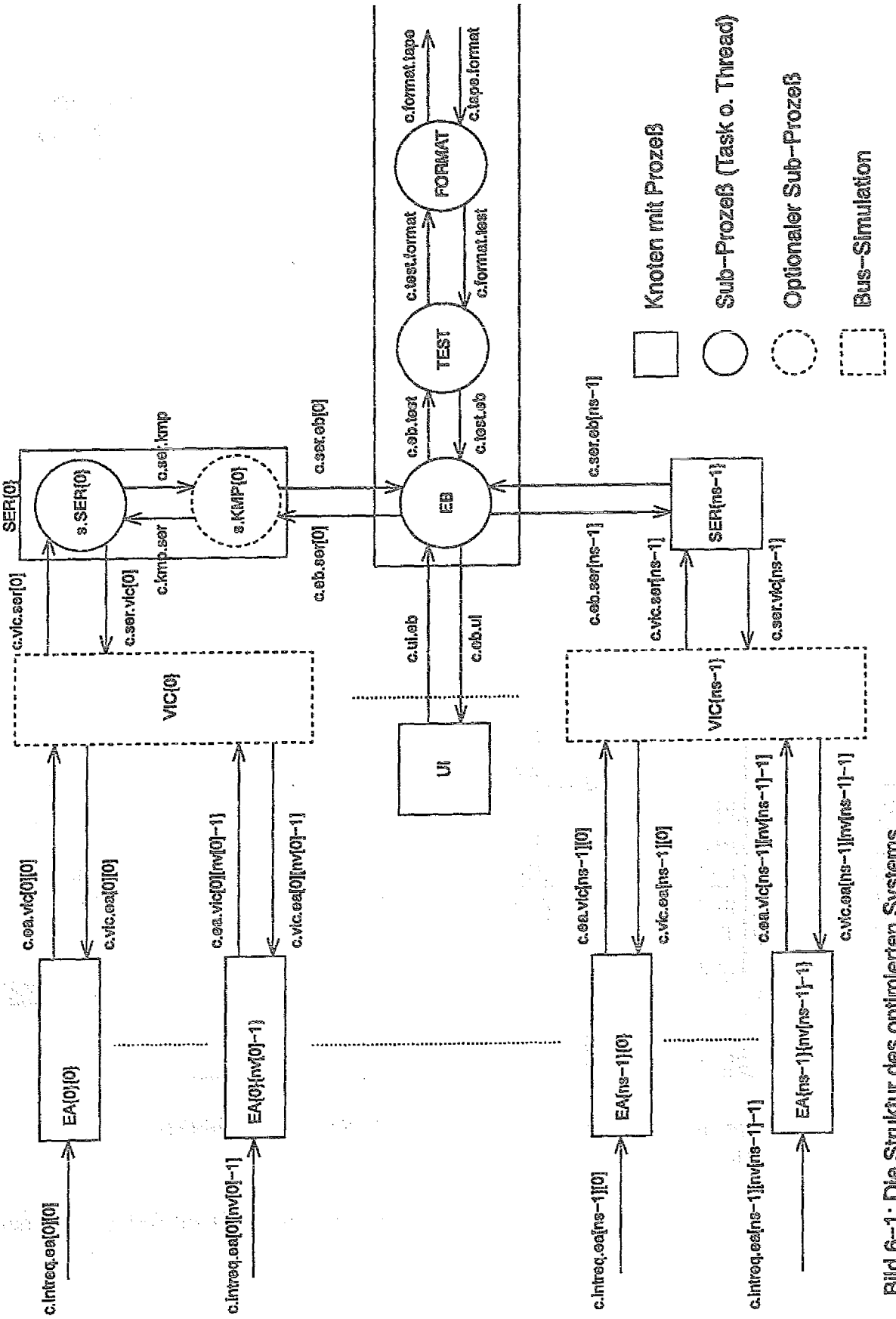


Bild 6--1: Die Struktur des optimierten Systems

6.2. Die optimierte Hardware

Wie ist es nun möglich, die Struktur von Bild 6-1 zu realisieren?
Aus dem Bild 6-1 ergibt sich, daß jeder SER{i}-Prozeß auf einem eigenen Prozessor ablaufen muß, damit die Datenübertragung in den VICbus-Strängen parallel zueinander ablaufen kann. Damit direkte Verbindungen zu dem EB-Prozeß geschaffen werden, müssen die SER{i}-Prozesse und der EB-Prozeß in Mikroprozessoren untergebracht werden, die nicht nur Rechenleistung, sondern auch Kanäle zur Interprozessorkommunikation bereitstellen.
Der bekannteste Typ solcher Prozessoren ist der Transputer wie z. B. der T800 oder der T9000 der Firma inmos /INMO 89/ /INMO 91/ /INMO 92.1-3/.

6.2.1. Auswahl des Prozessors

Bei dem System nach Bild 4-1 werden die SER{i}-Prozesse und der EB-Prozeß auf Motorola-CPUs zum Ablauf gebracht. Die erste Forderung, die sich daraus bei der Suche nach einer geeigneteren CPU ergibt, ist: Die Rechenleistung der gesuchten CPU sollte besser sein als die eines 68040-Mikroprozessors von Motorola. Die zweite Forderung verlangt nach einer großen Kommunikationsbandbreite und einer großen Anzahl von Kanälen zur Interprozessorkommunikation. Stellt man die Daten geeigneter CPUs in einer Tabelle dar, erhält man Bild 6-2:

CPU	Hersteller	MIPS S=Spitzenwert	Kommunik.-bandbreite in Mbyte/s	Kanal-anzahl	Verfügbar?
68030	Motorola	>25	0	0	Ja
68040	Motorola	>33	0	0	Ja
T805	inmos	30(S)	9,4	4	Ja
T9000	inmos	200(S)	80	4+2	Nein
TMS320C40	Texas Instruments	160(S)	40	6	Ja
PowerPC 601 + T805	Motorola + Texas Instruments	160(S)	9,4	4	1. Quartal 94

Bild 6-2: Leistungsdaten geeigneter CPUs nach /INMO 91/ /INMO 92/ /TI 91.1/ /PARS 94/.

Aus der Tabelle ergibt sich, daß der Signalprozessor TMS320C40 (oder kurz: der C40) heute (Juli 1994) der beste Prozessor zur Realisierung des Systems ist.

Der T805 scheidet aufgrund seiner geringen Kommunikationsbandbreite aus, und der T9000 war bis zum dritten Quartal 1993 nicht verfügbar.
Das Modul der Firma Parsytec, das die hohe Rechenleistung eines PowerPC 601 Mikroprozessors mit der Kommunikationsfähigkeit eines T805-Transputers vereint, ist für diese Anwendung aufgrund der geringen Kommunikationsbandbreite und der fehlenden parallelen Speicherschnittstelle nicht verwendbar.
Die Entscheidung, den T9000 nicht einzusetzen, stellte sich als richtig heraus, denn im zweiten Quartal 1994 war es immer noch nicht möglich ein fehlerfreies Exemplar zu bekommen.
Für die Motorola-Prozessoren ist eine sinnvolle Angabe der Rechenleistung in der Einheit MIPS (Million Instructions Per Second) schwierig, da es sich um CISC-Prozessoren handelt (CISC = Complex Instruction-Set Computer). Hier ist eine untere Grenze angegeben, die sich aus den üblichen Taktfrequenzen ergibt.

Der C40 hat folgende Eigenschaften:

- Zwei externe 32 bit breite Daten- und Adreßbusse, mit einer Transferrate bis zu 2 x 40 Mbyte/s
- Sechs Kommunikationsschnittstellen für die Interprozessorkommunikation: 8 bit parallel, je 20 Mbyte/s
- Sechskanal DMA-Koprozessor (Direct Memory Access-Koprozessor) unabhängig von der CPU
- Hochleistungs Signalprozessor-CPU: 40 MIPS
- 512 byte Cache und 8 kbyte RAM Speicherplatz auf dem Prozessor-Chip
- JTAG (Joint Test Action Group) -Schnittstelle zu einem Analysemodul auf dem Prozessor-Chip
- Vier interne Busse: Programm-, Daten 1-, Daten 2- und DMA-Koprozessor-Bus

Die technischen Daten des C40 sind in /TI 91.1/ teilweise ungenau angegeben. So könnte man aus den obigen Daten schließen, daß der C40 eine Kommunikationsbandbreite von 6 x 20 Mbyte/s = 120 Mbyte/s hat. Untersuchungen von /BORG 93/ haben jedoch ergeben, daß die Bandbreite des DMA-Koprozessors nur 40 Mbyte/s ist. D. h.: Werden zwei Kommunikationskanäle mit 20 Mbyte/s bedient, dann kann der DMA-Koprozessor keine weiteren Aufgaben erfüllen!

Die Artikel: /ANDE 92/, /ANDR 90/, /ANDR 92.1/, /ANDR 92.2/ und /WIES 92/ beschäftigen sich mit weiteren Einsatzmöglichkeiten für Signalprozessoren. /WILS 92/ beschreibt, daß sich RISC-CPUs (RISC = Reduced Instruction-Set Computer) und Signalprozessoren sehr ähneln. Er kommt auch zu dem Ergebnis, daß Signalprozessoren beim Verarbeiten von Datenströmen vergleichbaren RISC-

CPUs überlegen sind. Daher werden in Bild 6-2 keine reinen RISC-CPUs betrachtet.

6.2.2. Auswahl des lokalen Busses

Schon bei dem in Bild 4-1 beschriebenen System hat man das Problem, die Systemkomponenten mit den verschiedenen Busstandards zu einer Einheit zu verschmelzen. Folgende Schnittstellen (<=>) werden benötigt:

- CAMAC zu VICbus, realisiert durch die Schnittstellen:
  CAMAC <=> Motorola 680x0 Bus <=> VICbus /ERVE 92/

- FASTBUS zu VICbus, realisiert durch die Schnittstellen:
  FASTBUS <=> Motorola 680x0 Bus <=> VICbus /STRU 92/

- VICbus zu VMEbus, realisiert durch die Schnittstellen:
  VICbus <=> VSB <=> Motorola 680x0 Bus <=> VMEbus /CES 93/ /HOLZ 92.2/

- VMEbus zu SCSI, realisiert durch die Schnittstellen:
  VMEbus <=> Motorola 680x0 Bus <=> SCSI /CES 93/

- VMEbus zu TURBOchannel, realisiert durch die Schnittstellen:
  VMEbus <=> Motorola 680x0 Bus <=> VSB <=> VICbus <=> TURBO-channel /HOLZ 92.1/ /HOLZ 93/

Man hat also Standards verwendet, wann immer dies möglich war. Nur der Motorola 680x0 Bus fällt aus diesem Rahmen. Es war aber notwendig, einen lokalen Bus oder einen "gemeinsamen Nenner" zu wählen, um das Entwickeln neuer Schnittstellen zu vereinfachen. Da für den Motorola 680x0 Bus schon einige Entwicklungen abgeschlossen waren, wurde dieser für die COSY-Experimentausrüstung gewählt.
Daraus ergibt sich aber folgender Nachteil: Da der Motorola 680x0 Bus kein standardisierter Bus ist, hat man auch in Zukunft viel Entwicklungsarbeit zu leisten, um neue Schnittstellen zu realisieren. Dies wird notwendig, wenn modernere CPUs mit neuen Prozessorbussen, oder neue standardisierte Systembusse in das bestehende System integriert werden sollen. Außerdem ist dieser Prozessorbus für die 680x0-er Familie optimiert, und nicht auf die Anforderungen des schnellen Datentransfers in einem Multiprozessorsystem.

Es gibt also für den Motorola 680x0 Bus keine Vorschriften im Sinne einer Busspezifikation, d. h. es sind zwar Zeitabläufe spezifiziert, es fehlen aber genaue

elektrische und mechanische Standards. Es fehlt auch eine einheitliche Programmierschnittstelle.
Daher ergibt sich die Forderung, auch für den lokalen Bus einen Standard zu verwenden. Die Wahl fiel auf den PCI-Standard (Peripheral Component Interconnect-Standard), der in der PC-Welt innerhalb eines Jahres schon weite Verbreitung gefunden hat und eine zukunftsorientierte Spezifikation besitzt /HUTH 93/ /INTE 93/ /MAIE 93/ /NASS 93/ /SCHN 93.1/ /SCHN 93.2/ /SCHN 94.1/.

6.2.2.1. Eigenschaften des PCI-Busses

- Prozessorunabhängig
- Gemultiplexte Version des intel486 Busses, mit zusätzlichen Kontrollmechanismen, modifiziert und erweitert für optimalen Ein- und Ausgabe-Support
- Schneller Schreib- und Lese-Modus durch Blocktransfers
- Synchroner Bus mit Taktrate von 20 bis 33 MHz
- Hohe Transferraten:
  120 Mbyte/s (132 Mbyte/s Spitzenwert) bei 32 bit Busbreite
  240 Mbyte/s (264 Mbyte/s Spitzenwert) bei 64 bit Busbreite
- Volle parallele Operation zum Prozessor- und Speicher-Subsystem
- Multi-Master System
- Versteckte (überlappende) zentrale Arbitrierung
- Drei physikalische Adreßbereiche: Speicher, Ein- und Ausgabe, Konfiguration
- Einfacher Übergang zur Niedervolt-Technologie (3,3 V - statt 5 V - Logik)

6.2.3. Zu entwickelnde Schnittstellen

Im Vergleich zu der Aufstellung in Kapitel 6.2.2 werden für das optimierte System folgende Schnittstellen benötigt: (Die noch zu entwickelnden sind durch **Fettdruck** markiert)

- CAMAC zu VICbus, realisiert durch die Schnittstellen:
  CAMAC <=> Motorola 680x0 Bus <=> VICbus /ERVE 92/

- FASTBUS zu VICbus, realisiert durch die Schnittstellen:
  FASTBUS <=> Motorola 680x0 Bus <=> VICbus /STRU 92/

- VICbus <=> C40-Bus, realisiert durch die Schnittstellen:
  VICbus <=> PCI-Bus /HAGE 93/ und **PCI-Bus <=> C40-Bus** (Kapitel 8.4)

- PCI-Bus zu SCSI, realisiert durch einen ASIC (Application Specific Integrated Circuit) /NCR 93/

- C40-Bus zu Hauptrechner-CPU, realisiert durch die Schnittstellen:
  C40-Bus <=> PCI-Bus (Kapitel 8.4) und PCI-Bus <=> Hauptrechner-PCI-Bus (ASIC vorhanden: /DEC 93/)

Die PCI-Bus <=> C40-Bus Schnittstelle ist in Kapitel 8.4 beschrieben.

6.3. Auswahl der Programmierumgebung für den C40

In Kapitel 6.2.1 wird verdeutlicht, warum der C40 als CPU gewählt wurde und in Kapitel 5 ist das Datenerfassungsproblem in occam 2 beschrieben worden. Es gibt aber für diesen Signalprozessor keinen occam 2 Übersetzer (Compiler). Also muß eine andere Programmiersprache mit passender Entwicklungsunterstützung gefunden werden.
Für die Softwareunterstützung eines Multiprozessorsystems mit C40 Prozessoren und Interprozessorkommunikation, wie es das fertige Datenerfassungssystem benötigt, gibt es drei Möglichkeiten: SPOX, ein Echtzeit Signalprozessor-Betriebssystem /SPEC 91/; Helios, ein Echtzeit Multiprozessor-Betriebssystem /DIST 91/; und den parallel C Compiler von 3L /3L 92/, ein Compiler der mit Hilfe eines Echtzeitkerns zu einem Minimal-Betriebssystem wird. Helios und parallel C von 3L sind ursprünglich für den Transputer entwickelt worden. Daher findet man hier auch das Konzept der kommunizierenden sequentiellen Prozesse (CSP) /HOAR 78/.
Die Evaluierung ergab Vorteile für den Einsatz von parallel C. Denn: SPOX ist mit 30 TDM zu teuer und Helios mit etwa 1 Mbyte Speicherbelegung pro Knoten (Node) zu umfangreich. Der Preis für parallel C ist mit ungefähr 8 TDM relativ gering, der Speicherbedarf mit < 100 kbyte akzeptabel.
Das Ergebnis ist somit, daß parallel C von 3L die beste käufliche Softwareunterstützung für das optimierte System bietet.

Die wichtigsten Eigenschaften von parallel C sind:

- Routinen zur Interprozeßkommunikation nach dem Prinzip der kommunizierenden sequentiellen Prozesse (CSP)
- Multitasking Unterstützung
- Multiprocessing Unterstützung
- Echtzeit-Mikrokern mit Prozeßmanager und Interrupt-Bearbeitung
- Einfache Abbildung des parallelen Kodes auf eine gegebene Hardware-Umgebung
- Abbildung auf eine andere Hardware-Umgebung durch Umschreiben einer Konfigurationsdatei ohne erneute Übersetzung (Compilierung) durchführbar
- Umschreiben eines occam 2 Programms in parallel C von 3L möglich

6.3.1. Beispiel für die Übersetzung: occam 2 zu parallel C

Es folgt ein Beispiel, das veranschaulicht, wie ein einfaches occam 2 Programm in parallel C umzuschreiben ist.

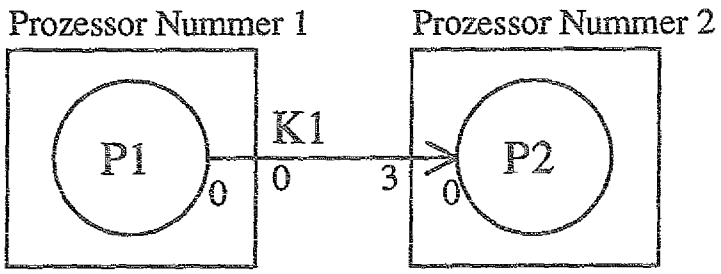


Bild 6-3: Zwei kommunizierende sequentielle Prozesse P1 und P2

Seien P1 und P2 zwei sequentielle Prozesse, und K1 der sie verbindende Kanal wie in Bild 6-3, dann kann das Bild z. B. in folgendes occam 2 Programm umgesetzt werden:

```
CHAN OF INT::[]INT K1:
PLACED PAR
  PROCESSOR Nummer.1      -- Folgender Kode wird in Prozessor
                          -- Nummer 1 plaziert
    PLACE K1 AT 0:        -- Zuordnung K1 -> Anschluß 0
    [256] INT dat1:       -- Lokale Daten für Sequenz 1
    WHILE TRUE
      SEQ
        P1()                -- Erzeugerprozeß
        K1 ! 256::dat1      -- Ausgabe von 1024 byte auf K1
  PROCESSOR Nummer.2      -- Folgender Kode wird in Prozessor
                          -- Nummer 2 plaziert
    PLACE K1 AT 3:        -- Zuordnung K1 -> Anschluß 3
    [256] INT var1:       -- Lokale Daten für Sequenz 2
    INT var.size:         -- dito.
    WHILE TRUE
      SEQ
        K1 ? var.size::var1  -- Einlesen von K1
        P2()                 -- Verbraucherprozeß
```

Das Programm generiert zwei parallel ablaufende Sequenzen. In der ersten Sequenz ist P1 ein Erzeugerprozeß, der das eindimensionale Feld "dat1" mit 256 - 32 bit Worten füllt. Danach wird versucht, diese Daten über den Kanal K1 auszugeben.
Erst wenn in der zweiten Sequenz der Leseprozeß: "K1 ? var.size::var1" aktiv wird, findet die Datenübertragung statt und die Prozesse sind für diese Zeit synchronisiert. Dabei wird in "var.size" der Wert 256 gespeichert.

Üblicherweise erfolgt die Entwicklung eines occam-Programms auf einem Prozessor mit logischen Kanälen (Verbindungen). Ist der Kode fehlerfrei, kann er auf mehrere Prozessoren, mit physikalischen Kanälen (Leitungen) und logischen Kanälen, abgebildet werden. Die Schlüsselwörter PLACED PAR, PROCESSOR *Nummer.i* und PLACE *Logischer.Kanal* AT *Kommunikationsschnittstelle.i*: dienen dazu, ein occam 2 Programm auf eine konkrete Hardware-Architektur abzubilden. Durch PLACED PAR und PROCESSOR Nummer.1 wird der folgende Kode in den Prozessor mit der Nummer 1 geladen. Es erfolgt also die Abbildung eines logischen Prozesses auf einen Prozessor. Durch PLACE K1 AT 0: wird der Kanal K1 mit der Kommunikationsschnittstelle 0 verbunden. Der logische Kanal K1 wird also mit dem Anschluß 0 des Prozessors verknüpft. In occam 2 kann also der Kode für ein kleineres Projekt in einer Datei übersichtlich aufgeschrieben werden. Beim Bearbeiten größerer Projekte wird man natürlich die Programme, zur besseren Übersicht, in mehreren Dateien unterbringen.
Die oben beschriebene Methode hat den Nachteil, daß der Kode bei der Anpassung an eine andere Hardware umgeschrieben und neu compiliert werden muß.

In parallel C muß dieses Programm durch drei Dateien formuliert werden. Man legt für jeden logischen Prozeß (Task) eine Datei an und erstellt noch eine Konfigurationsdatei, die die Abbildung des Programms auf die Ziel-Hardware beschreibt. Das hat den Vorteil, das bei der Anpassung an eine andere Hardware lediglich diese Datei verändert werden muß. Ein neue Übersetzung (Compilation) ist nicht erforderlich. Dies erleichtert besonders die Systementwicklung auf einem Prozessor.

Datei producer.c:

```
/* Erzeugerprozeßsequenz */
include<chan.h>

main (int argc, char *argv[], char *envp[],
      CHAN *outK1[], int outs);
{
   int dat1[256];
   dat1.size = 256;
   for(;;){
     P1();    /* Erzeugerprozedur */
     chan_out_word(&dat1.size, outK1[0]);
     chan_out_message(dat1.size, &dat1[0], outK1[0]);
   }
}
```

Datei consumer.c:

```
/* Verbraucherprozeßsequenz */
include<chan.h>

main (int argc, char *argv[], char *envp[],
      CHAN *inK1[], int ins);
{
   int var1[256], var.size;
   for(;;){
     chan_in_word(&var.size, inK1[0]);
     chan_in_message(var.size, &var1[0], inK1[0]);
     P2();     /* Verbraucherprozedur */
   }
}
```

Datei config.cfg:

```
! = Kommentar
!Hardware-Konfiguration
processor Nummer1
processor Nummer2

!Physikalische Kanäle (Leitungen)
wire ? Nummer1[0]   Nummer2[3]

!Software-Konfiguration
task producer ins=0 outs=1
task consumer ins=1 outs=0

place producer Nummer1
place consumer Nummer2

!Logische Kanäle (Verbindungen)
!connection ? Ausgang[i] Eingang[j]
connection ? producer[0] consumer[0]
```

Man erkennt, daß die Notation des Programms in parallel C wesentlich schwerer zu lesen und damit auch schwerer zu schreiben und zu verstehen ist. Der Grundgedanke von occam wird also verletzt, aber es gibt keine bessere Alternative.
Die Deklaration der Kanäle erfolgt durch Zeiger auf die Struktur CHAN, welche in der Datei chan.h definiert ist. Die occam 2 Anweisung "K1 ! 256::dat1" muß durch die beiden Prozeduraufrufe chan_out_word(...) und chan_out_message(...) formuliert werden. Analoges gilt für die Verbraucherdatei consumer.c.
Um die Programme nach dem Übersetzen und Binden in zwei Prozessoren laden zu können, ist die Konfigurationsdatei config.cfg erstellt worden. Die Datei bedeutet im Klartext: Es gibt zwei Prozessoren Nummer1 und Nummer2. Die KS 0 (Kommunikationsschnittstelle 0) von Nummer1 ist mit der KS 3 von Nummer2 durch eine Leitung (wire) verbunden. Es gibt zwei Prozesse: producer und consumer. Der producer besitzt einen Ausgang (output-port): ins=0 outs=1,

der consumer einen Eingang (input-port): ins=1 outs=0. Der producer-Prozeß soll in Prozessor Nummer1, der consumer-Prozeß in Prozessor Nummer2 ablaufen. Der logische Ausgang 0 des producers ist über einen logischen Kanal (connection) mit dem logischen Eingang 0 des consumers verbunden.

Die Methodik dieses Beispiels kann verwendet werden, um die Struktur nach Bild 5-2 in parallel C abzubilden.

6.4. Beschreibung der optimierten Architektur

In den Kapiteln 4 und 5 wurde eine Strukturanalyse des Datenerfassungssystems für COSY 11 durchgeführt, indem die vorhandene Hard- und Software durch Strukturbilder abstrahiert wurde. In den Kapiteln 6.1 und 6.2 wird diese Struktur verbessert, und das Ergebnis in Bild 6-1 dargestellt.
Für die Implementation dieser Struktur ist nun der umgekehrte Weg einer Systemsynthese zu gehen.
Aus der Aufstellung in Kapitel 6.2.3 und Bild 6-1 ergibt sich, daß ein Modul (eine Systemkomponente) zu entwerfen ist, das ein C40-Mikroprozessorsystem mit einer PCI-Schnittstelle enthält. Auf diesem C40-Modul soll sowohl ein SER{i}-Prozeß als auch der EB-Prozeß ablaufen können.
Bei der Verwendung des C40-Moduls als Subeventreader muß eine Verbindung zu einer VICbus <=> PCI-Bus Schnittstelle und zu mindestens einer C40-Kommunikationsschnittstelle ermöglicht werden.
Nutzt man das gleiche C40-Modul als Eventbuilder, so muß eine Verbindung zu dem Hauptrechner, dem SCSI, und möglichst vielen C40-Kommunikations-schnittstellen realisiert werden.
Ideal wäre also ein Baukasten von Modulen, die sich wie in Bild 6-4 dargestellt konfigurieren lassen (vgl. rechte Hälfte von Bild 6-1).

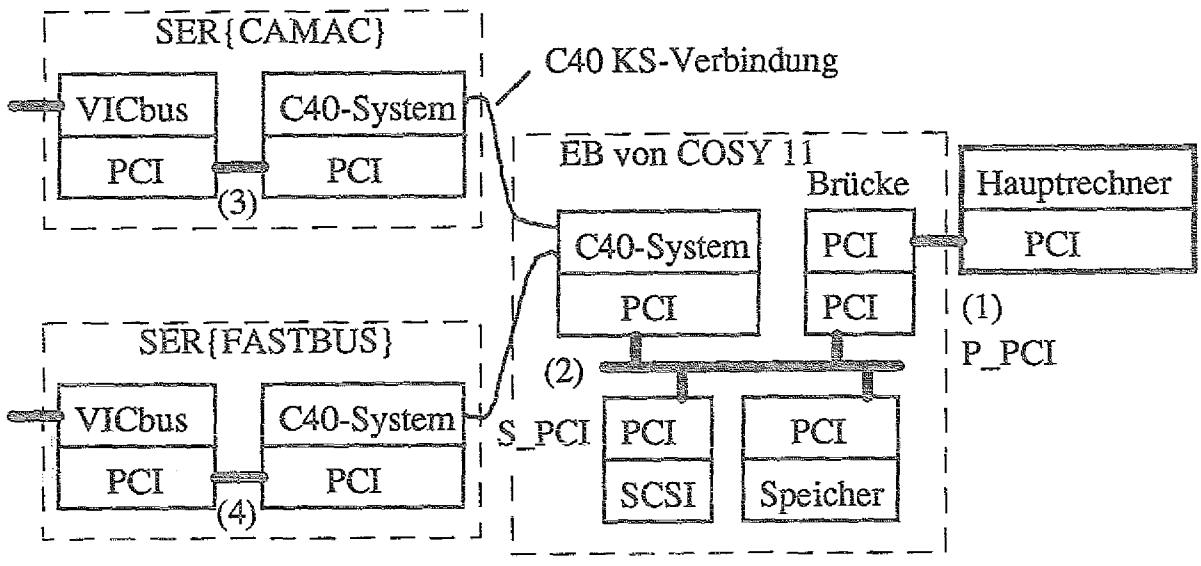


Bild 6-4: Hardware-Module des verbesserten Systems für COSY 11

Da die Datenübertragungsrate der C40-Kommunikationsschnittstellen von der Kabellänge abhängig ist (vgl. 6.4.1, Zu 2), wäre es ideal, alle Komponenten nach Bild 6-4 in dem Gehäuse eines EISA/PCI-PCs unterzubringen. Aufgrund des hohen Hardware-Aufwands benötigt man aber für dieses System zwei PCI-Steckplätze und vier PCI- oder EISA-Steckplätze. Die meisten PCs mit PCI als lokalem Bus besitzen aber nur drei PCI-Steckplätze und weniger als vier EISA-Steckplätze für EISA-Karten mit der maximal erlaubten Länge, d. h. selbst wenn es gelingen würde, das ganze System in einem PC-Gehäuse unterzubringen, würden die Karten nicht mehr ausreichend gekühlt. Die Lösung dieses Problems liegt in der Verwendung einer externen Box, die die Subeventreader, einen Kühlventilator und einen preiswerten Standardbus (ISA) zur Spannungsversorgung und zur mechanischen Fixierung enthält. Dabei müssen aber die Kabel zwischen PC-Gehäuse und externer Box möglichst kurz gehalten werden. Im PC-Gehäuse verbleibt lediglich der Eventbuilder in einem PCI-Steckplatz. Da dieser Steckplatz nur mit einer kapazitiven Last von maximal 10 pF belegt werden darf /INTE 93/, ist die Verwendung eines Brückenbausteins zur kapazitiven und systematischen Entkopplung zwingend.
Damit hat man vier verschiedene PCI-Bussegmente im System. Das Segment Nr. 1 ist der primäre Bus des PCI-Hauptrechners (P_PCI). Das Segment Nr. 2 ist der über eine PCI/PCI-Brücke angeschlossene sekundäre Systembus (S_PCI) oder Eventbuilder-Systembus.
Die Segmente Nr. 3 und Nr. 4 sind lokal und werden komplett von den C40-Systemen in den Subeventreadern gesteuert.
Der Speicher am Segment Nr. 2 ist ein 4 Mbyte großes DRAM (Dynamic Random Access Memory), das als makroskopischer Puffer für die Event-Datenpakete dienen soll. Das C40-System kann in seinem eigenen Speicher einen

Event zusammenstellen, während die SCSI-Schnittstelle früher abgelegte Events **eigenständig** auf das Magnetband schreibt.
Die Systemkomponenten in Bild 6-4 müssen nun auf verschiedene Karten verteilt werden. Die Forderung nach der Wiederverwendbarkeit des C40-Systemmoduls und der räumliche Platzbedarf der Bauteile lassen eine Aufteilung nach Bild 6-5 zu.

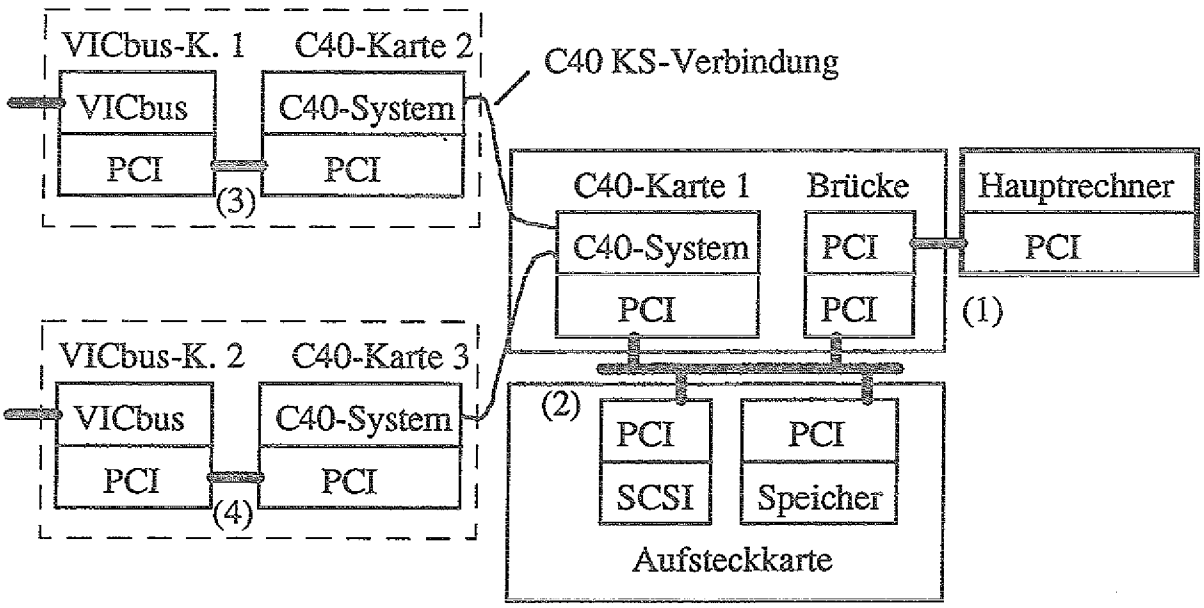


Bild 6-5: Leiterplatten des verbesserten Systems für COSY 11

Die Beschreibung der verschiedenen Karten folgt in den nächsten Kapiteln.

6.4.1. Die C40-Karte

Nach Bild 6-5 und Bild 6-7 muß die C40-Karte folgendes enthalten:

1. C40-Mikrorechnersystem
2. C40-Kommunikationsschnittstellen-Treiber
3. C40-Bus <=> PCI-Bus Schnittstelle
4. PCI-Bus Treiber
5. Deaktivierbare zentrale Funktionen für den PCI-Bus
6. Deaktivierbare PCI/PCI-Brücke
7. PCI-Kartenstecker
8. ISA-Kartenstecker
9. Stecker für PCI-Kabelverbindung
10. Stecker für PCI-Aufsteckkarte

Zu 1, C40-Mikrorechnersystem:

Die C40-Karte enthält ein Modul mit einem C40-Signalprozessor nach dem TIM-40 Standard (Texas Instruments Modul für Systeme mit dem C40) /TI 93.1/. Dieses Modul, MDC40S2 der Firma LSI (Loughborough Sound Images), besitzt 1 Mbyte Speicherplatz am lokalen C40-Bus und 0,5 Mbyte am globalen C40-Bus, ein Identifikations-ROM mit Konfigurationsdaten für den Prozessor und einen 40 MHz Taktgeber (Clock) /EDWA 92/ /LSI 93.1/. Es ist als Aufsteckkarte für ISA-, EISA-, oder VMEbus-Karten dimensioniert.
Die Verwendung dieses Moduls hat die Hardware-Entwicklung beschleunigt, da eine Leiterplatte mit diesen Basiselementen eines C40-Systems nicht mehr selbst entwickelt werden mußte. Ein Blockschaltbild des Moduls ist in Bild 6-6 dargestellt.

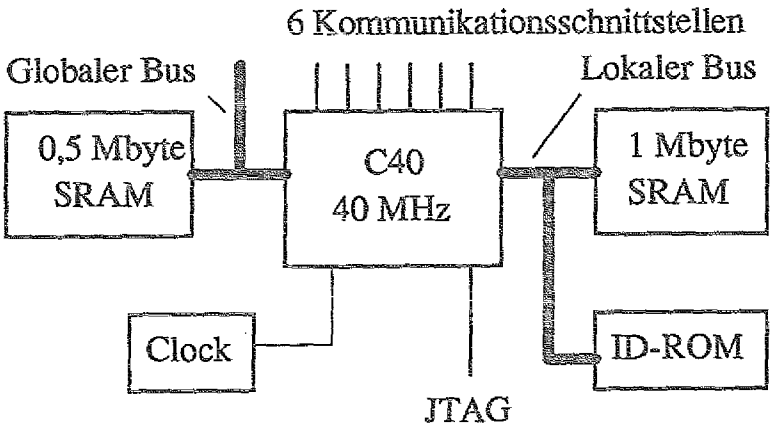


Bild 6-6 Blockschaltbild des MDC40S2 TIM-40 Moduls /LSI 93.1/

Die Verteilung des Speichers auf die beiden Busse des C40 erlaubt es, daß die CPU des C40 mit dem lokalen Speicher arbeiten kann, während der DMA-Koprozessor parallel dazu Daten über den globalen Bus transferiert und einige Kommunikationsschnittstellen bedient!

Zu 2, C40-Kommunikationsschnittstellen-Treiber:

Diese Treiber müssen bidirektional sein und durch eine einfache Logik automatisch umschalten. Es sind Schutzmechanismen vorzusehen, die es ausschließen, daß die Treiber zweier KS (Kommunikationsschnittstellen) gegeneinander arbeiten. Die Leitungen der KS des C40 sind direkt auf Stecker am TIM-40 Modul geführt. Deshalb braucht man auf der C40-Karte solche bidirektionalen Puffer, um die KS mit Kabeln bis zu 1,5 m Länge zu verbinden. Beträgt die

Kabellänge 1 m, so reduziert sich die Datenübertragungsrate auf etwa 10 Mbyte/s /LSI 93.3/.

Zu 3, C40-Bus <=> PCI-Bus Schnittstelle:

Ein PCI-Gerät sollte durch einen ASIC (Application Specific Integrated Circuit) mit dem PCI-Bus verbunden werden /INTE 93/. Für die Verbindung zum SCSI und für die Brücke (Bridge) zwischen einem sekundären PCI-Bus und einem primären Hauptrechner-PCI-Bus gibt es solche Bauelemente schon /DEC 93/ /NCR 93/.
Es fehlen also ASICs für die Schnittstellen: VICbus <=> PCI-Bus und C40-Bus <=> PCI-Bus. Da im Zentrallabor für Elektronik keine Ressourcen zur Verfügung stehen, solch komplexe ASICs herzustellen bzw. zu programmieren, mußten bei der Entwicklung dieser Schnittstellen mehrere einfachere ASICs verwendet werden. Dies ist in Kapitel 8.4 beschrieben.

Zu 4, PCI-Bus Treiber:

Diese Treiber müssen bidirektional sein und dem PCI-Standard entsprechen /INTE 93/. Im Standard wird auch die Generierung eines Paritäts-Bits gefordert. Dieses ließ sich am einfachsten durch besondere Treiber realisieren, die ein solches Bit intern generieren.

Zu 5, Deaktivierbare zentrale Funktionen für den PCI-Bus:

Ein Teil der C40-Bus <=> PCI-Bus Schnittstelle besteht aus einem Arbiter für den PCI-Bus und einer eigenen PCI-Bustaktgenerierung. Diese Teile werden auch zentrale Ressourcen genannt /INTE 93/. Diese Funktionen werden aber nur für die PCI-Bussegmente Nr. 3 und Nr. 4 benötigt (vgl. Bild 6-4 und Bild 6-5). Für das Segment Nr. 2 werden die zentralen Ressourcen der Brücke verwendet. Das bedeutet, wenn die C40-Karte an diesem Segment verwendet wird, müssen die eigenen zentralen Ressourcen deaktiviert werden.
Im Segment Nr. 1 liefern Teile des Hauptrechnersystems die Arbitrierungssignale und das Taktsignal.

Zu 6, Deaktivierbare PCI/PCI-Brücke:

In den Segmenten Nr. 3 und Nr. 4 (vgl. Bild 6-4 und Bild 6-5) muß die Brücke deaktiviert werden, da keine Schnittstelle auf der primären Seite angeschlossen ist. Das bedeutet, daß die Brücke nicht konfiguriert werden kann /DEC 93/.

Bei der Verwendung als Eventbuilder steckt die Karte im PCI-Bus, und der C40 ist über eine PCI/PCI Brücke (Bridge) mit dem primären PCI-Bus des PCs verbunden. Der sekundäre PCI-Bus der Brücke verbindet das Eventbuilder C40-System, die SCSI-Schnittstelle und einen Pufferspeicher miteinander. Man hat so eine schnelle Verbindung zwischen dem PCI-PC und diesen Komponenten.

Zu 7,8,9 und 10, Steckerverbindungen:

Soll die C40-Karte mit dem Hauptrechner verbunden werden, so muß dies über einen PCI-Kartenstecker erfolgen. Die Verbindung zur Aufsteckkarte wird durch einen Stecker auf der Lötseite der Leiterplatte hergestellt.
Wird die C40-Karte als Subeventreader eingesetzt, dann befindet sie sich in einer Box mit ISA-Steckplätzen, und braucht daher auch einen ISA-Kartenstecker. Ein weiterer Stecker, auf der Bestückungsseite, wird dann für die PCI-Kabelverbindung benutzt.

Mit Hilfe des in 1. beschriebenen Moduls läßt sich das Blockschaltbild der C40-Karte wie in Bild 6-7 darstellen.

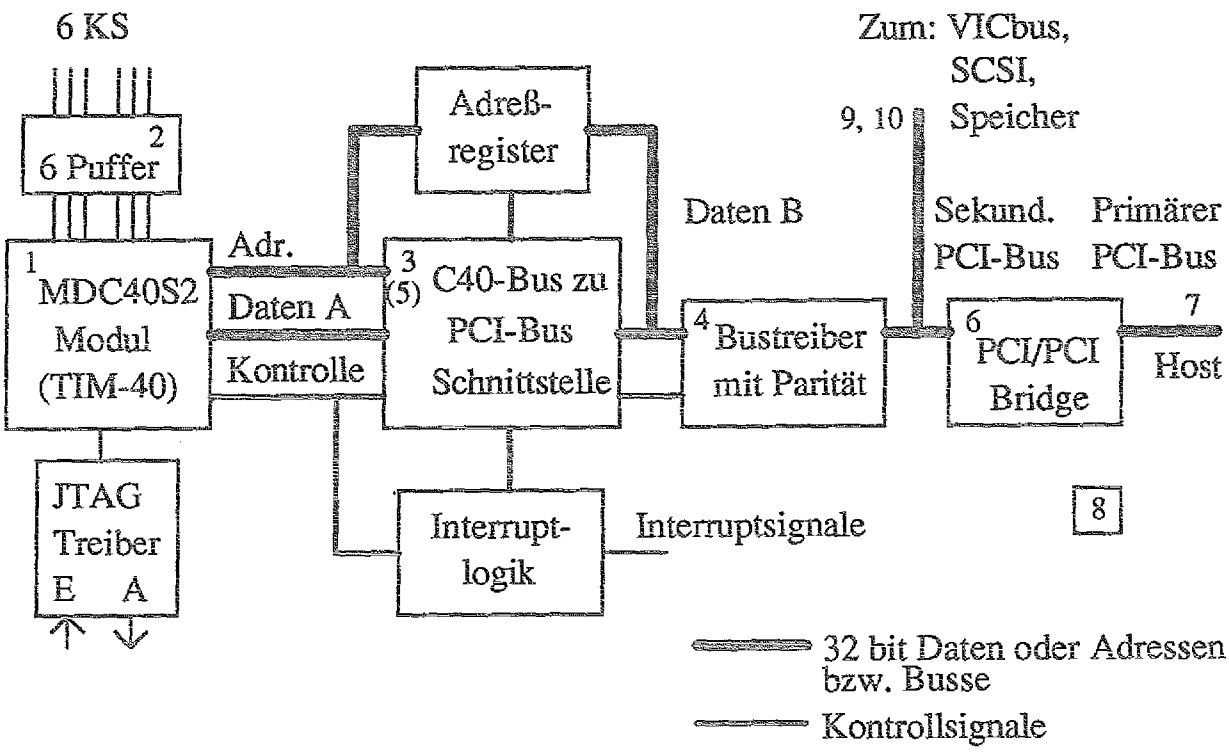


Bild 6-7 Blockschaltbild der C40-Karte

Für die JTAG-Schnittstelle sind ebenfalls einige Puffer zur Signalaufbereitung notwendig. Zwei Stecker (JTAG-Eingang: E und JTAG-Ausgang: A) erlauben die

Verkettung mehrerer Karten und damit die Störungssuche (debugging) in einem Multiprozessorsystem. Die C40-Bus zu PCI-Bus Schnittstelle ist recht komplex, und wird in Kapitel 8.4 detaillierter beschrieben.
Der C40-Bus hat, im Gegensatz zum PCI-Standard, getrennte Daten- und Adreßbusse. Mit einem Adreßregister und Puffern muß daher ein Multiplex-Verfahren angewandt werden.
Wird die C40-Karte als Subeventreader eingesetzt, dann können, über die Interrupt-Logik (Unterbrechungslogik), Signale von der VICbus-Karte Interrupts für den Signalprozessor auslösen, oder der C40 kann - umgekehrt - auch Signale an die VICbus-Karte senden. Dies erlaubt eine schnelle Reaktion des C40 auf Signale eines Controllers am VICbus.
Der Speicherplatz eines Subeventreaders kann durch die Aufsteckkarte (vgl. 6.4.3) um 4 Mbyte erweitert werden. Die PCI-SCSI Schnittstelle wird dann nicht benutzt.

Wird die C40-Karte als Eventbuilder eingesetzt, dann dient die Interrupt-Logik (Unterbrechungslogik) der Übermittlung von Interrupts zwischen PCI-PC und dem Signalprozessor.

6.4.2. Die VICbus-Karte

Nach Bild 6-5 enthält die VICbus-Karte die VICbus zu PCI-Bus Schnittstelle.

In Bild 6-8 ist das Blockschaltbild der VICbus-Karte dargestellt.

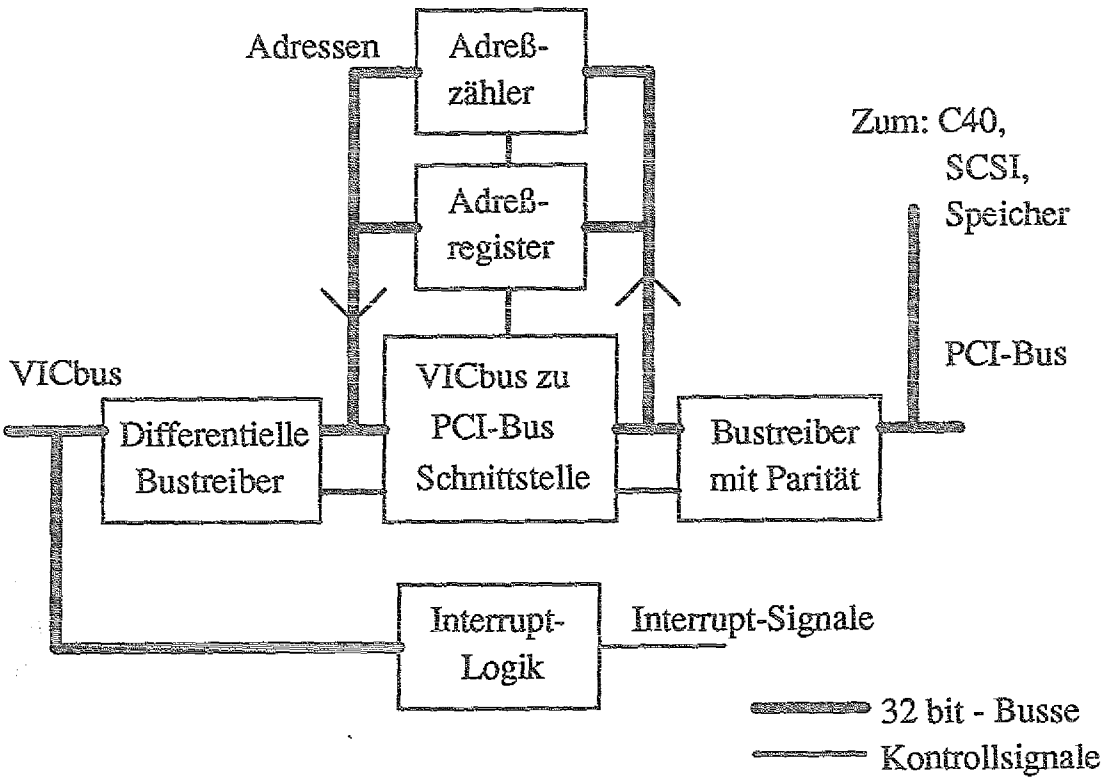


Bild 6-8: Blockschaltbild der VICbus-Karte

Der VICbus ist über differentielle Treiber, der PCI-Bus über Treiber mit integrierter Paritätsgenerierung mit den internen Bussen der VICbus-Karte verbunden. Die Interrupt-Logik (Unterbrechungslogik) formt Interrupt-Zyklen vom VICbus in interne Signale um oder generiert aus internen Interrupts entsprechende Zyklen auf dem VICbus.
Beim VICbus werden Daten und Adressen, wie beim PCI-Bus, über die gleichen Leitungen übertragen (Multiplexing). Im Fall eines Blocktransfers auf dem PCI-Bus werden zunächst eine Adresse und dann mehrere Datenwörter übertragen. Das ist auf dem VICbus auch so, jedoch ist die Größe eines Blocktransfers auf 2 kbyte begrenzt. Das bedeutet: Die Schnittstelle muß in der Lage sein, die fehlenden Adressen durch einen Zähler und ein Register, während eines laufenden Blocktransfers, selbst zu erzeugen. Diese Funktion, die Interrupt-Logik und die Steuerung der Schnittstelle erfordern komplexe Zustandsmaschinen und Bauelemente, und damit eine aufwendige Hardware /HAGE 93/.

6.4.3. Die Aufsteckkarte

Da auf der C40-Karte zuwenig Platz vorhanden war, mußten einige Funktionen in eine Aufsteckkarte integriert werden. Nach Bild 6-5 muß die Aufsteckkarte folgendes enthalten:

1. SCSI <=> PCI-Bus Schnittstelle

Die Aufsteckkarte enthält einen ASIC als Schnittstelle zum SCSI-Bus /NCR 93/, sowie einige konventionelle digitale Schaltkreise.

2. Speicher mit PCI-Bus Schnittstelle

Die Aufsteckkarte enthält auch einen Speicher mit PCI-Schnittstelle, der aus 4 Mbyte dynamischem RAM (DRAM) gebildet wird. Der PCI-Speicher ist notwendig, damit der ASIC von NCR als PCI-Master den Datentransfer durchführen kann, ohne die Arbeit des C40 zu behindern.

6.4.4. Das C40-Entwicklungssystem

Die Komponenten des Systems nach Bild 6-5 sind in Kapitel 6.4.1 - 6.4.3 beschrieben worden. In diesem Kapitel ist bisher nicht berücksichtigt worden, daß auch Software dazu notwendig ist, um das System zu betreiben. Es ist natürlich erforderlich, die Programme für die C40-Signalprozessoren zu laden, von Fehlern zu befreien und deren Ablauf und Parametrierung vom Hauptrechner aus zu steuern.

Der Hauptrechner soll mit dem Betriebssystem LynxOS arbeiten, weil es im CERN der kommende Standard für Echtzeitanwendungen verteilter Systeme werden wird /BLAK 94/ /PETE 92/.
LynxOS ist ein Echtzeitbetriebssystem und mit UNIX und POSIX kompatibel. Es ist außerdem auf viele Prozessoren portierbar (Motorola 68030, intel 386, intel 486, MIPS R3000, SUN sparc 2) /ECK 91/ /PETE 92/.
Für dieses System muß daher noch umfangreiche Treiber-Software für das Betriebssystem LynxOS erstellt werden.
Um die Entwicklung der Treiber und der C40-Software parallel durchführen zu können, wird ein Entwicklungssystem in einem zweiten PC, unter dem Betriebssystem MS-DOS, eingesetzt.

Das Bild 6-9 ist /LSI 93.2/ entnommen, und zeigt das Blockschaltbild der ISA-Karte für das Signalprozessorentwicklungssystem. Die anwendungsspezifischen Signalverbindungen sind wegen der besseren Übersichtlichkeit nicht dargestellt.

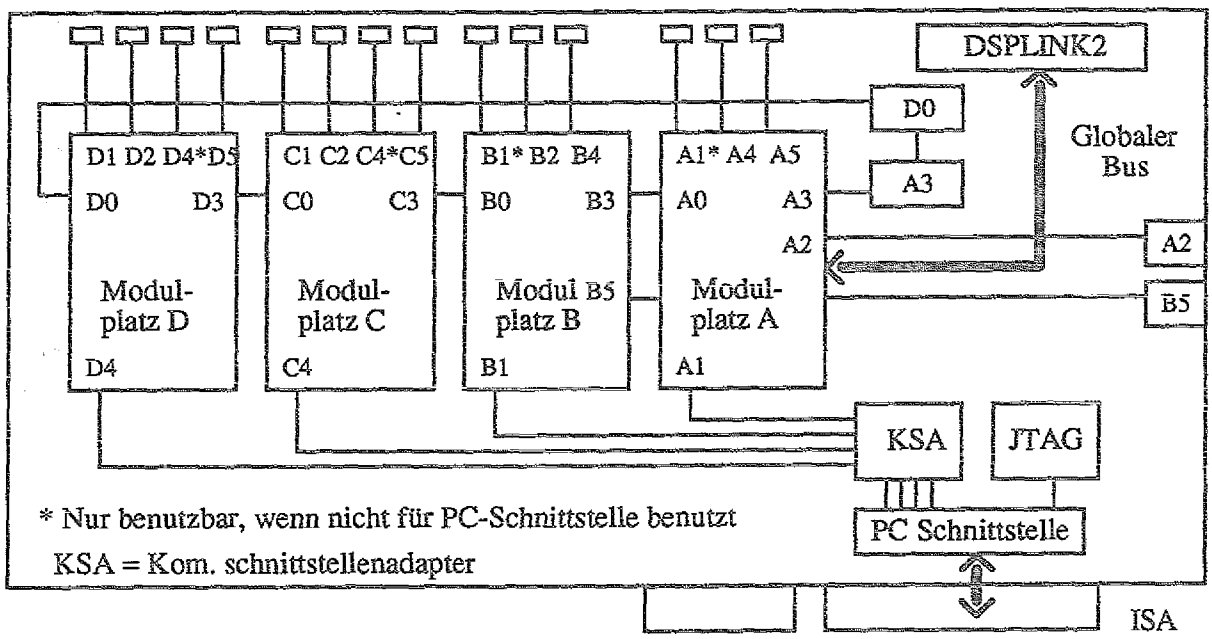


Bild 6-9: Das C40 Entwicklungssystem /LSI 93.2/

Die Hardware des Entwicklungssystems, oder die QPC/C40 Karte von LSI, besitzt vier Steckplätze für Module nach dem TIM-40 Standard. Pro Modul sind bis zu vier Kommunikationsschnittstellen über Puffer auf Stecker geführt, so daß viele Möglichkeiten für unterschiedliche Prozessornetzwerk-Topologien offen sind. Durch Verbinden der Stecker D0 und A3 sind die vier Prozessoren in einer Ringstruktur verbunden, ohne daß noch weitere Kabel angeschlossen werden müssen. Die PC-Schnittstelle wirkt über den Kommunikationsschnittstellen-adapter (KSA) auf bestimmte Kommunikationskanäle der Signalprozessoren.

Befindet sich diese Karte in einem PC unter MS-DOS, so können über einen Server (Dienstleistungsprozeß) der Firma 3L Programme für ein ganzes Netzwerk aus C40-Prozessoren geladen werden /3L 92/. Der Server stellt auch die Funktionen für Fehlersuche, Steuerung und Überwachung zur Verfügung. Der C40 kann auch aus einem Prozeß heraus durch den Server MS-DOS Befehle ausführen lassen, d. h. in einem C40-Programm können Standard C-Befehle wie printf(...) (formatiertes Drucken) oder scanf(...) (formatiertes Eingeben) benutzt werden!

Ein entsprechender Server unter dem Betriebssystem LynxOS existiert leider noch nicht.
Bei voller Bestückung kann diese Karte als Modell für ein Datenerfassungssystem mit drei Subeventreadern und einem Eventbuilder verwendet werden.
Die Karte verfügt auch über eine JTAG-Master Schnittstelle, mit einem TBC (Test Bus Controller) von Texas Instruments (74ACT8990 TBC), so daß mit passender Software ein Emulationssystem für C40-Multiprozessorsysteme zur Verfügung steht. Man kann also diese Karte über Kabel mit den JTAG-Schnittstellen eines realisierten Datenerfassungssystems verketten, so daß eine Fehlersuche im Zielsystem möglich wird /LSI 93.2/ /LSI 93.3/. Dabei können die TIM-40 Module des Emulationssystems auf den C40-Karten des Datenerfassungssystems wiederverwendet werden.
Diese hervorragenden Eigenschaften, und der relativ niedrige Preis (4.500 DM ohne TIM-40 Modul, im zweiten Quartal 1993), waren die Gründe für die Anschaffung gerade dieses Entwicklungssystems.

6.4.5. Komplette Architektur

Berücksichtigt man die bisher gemachten Überlegungen, so sieht das verbesserte Datenerfassungssystem für COSY 11 aus wie in Bild 6-10.

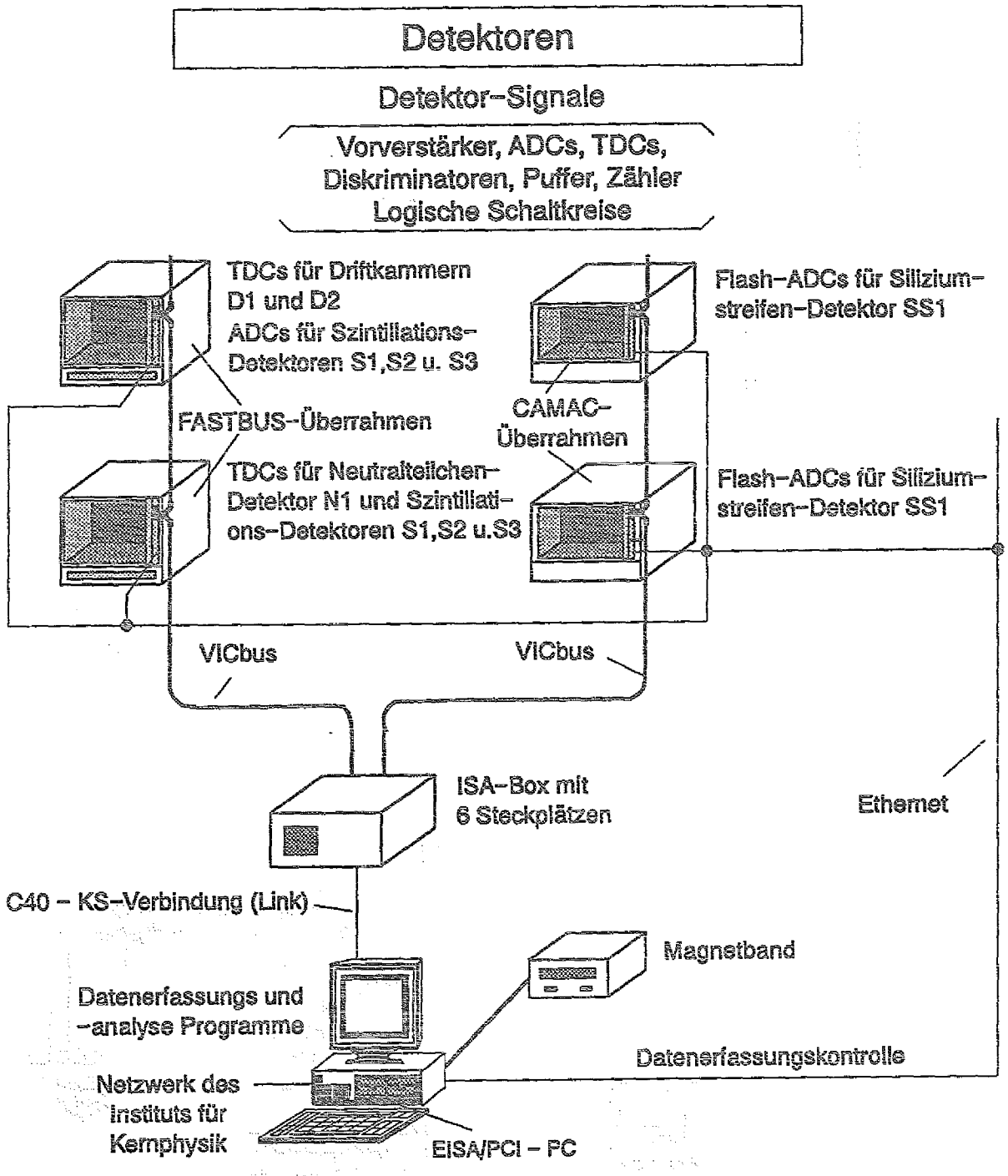


Bild 6-10: Aufbau des optimierten Datenerfassungssystems

Bringt man die Darstellung nach Bild 6-5 und Bild 6-10 in einem gemeinsamen Bild unter, so erhält man Bild 6-11.

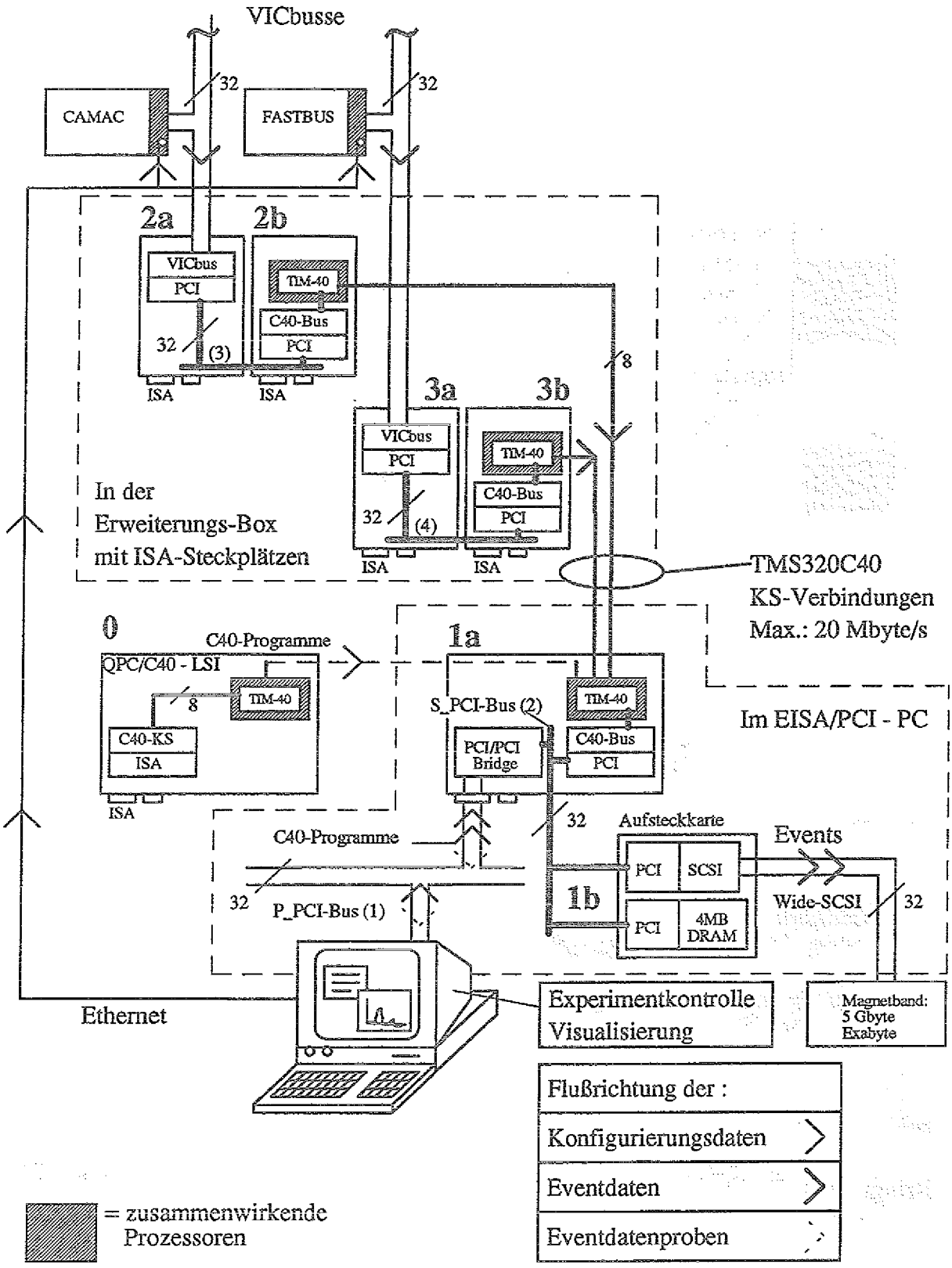


Bild 6-11: Blockschaltbild und Aufbau des optimierten Datenerfassungssystems

Die in Bild 6-1 aufgezeichneten Prozesse lassen sich nun auch auf Bild 6-10 und Bild 6-11 abbilden. Die Eventaquisitions-Prozesse: EA{i}{j} laufen in den Controllern der CAMAC- und FASTBUS-Überrahmen ab. Der Datentransport zu den beiden Subeventreadern erfolgt, wie auch in Bild 4-1, über zwei VICbus-Stränge.

Eine VICbus-Karte (2a oder 3a) und eine C40-Karte (2b oder 3b) bilden zusammen die Hardware, die für die Funktionalität eines Subeventreaders (2a und 2b oder 3a und 3b) notwendig ist.
Eine C40-Karte (1a) und eine Aufsteckkarte (1b) realisieren die Hardware eines Eventbuilders.

Es ist eine Option des Systems, die Speicherkapazität eines Subeventreaders zu erhöhen, indem auch ein Subeventreader mit einer Aufsteckkarte versehen wird. Die SCSI-Schnittstelle wird dann nicht benutzt.

Die verschiedenen PCI-Bus-Segmente sind hier, analog zu den Bildern 6-4 und 6-5, mit den Ziffern (1), (2), (3) und (4) bezeichnet.

Die Subeventreader-Prozesse: SER{i} sind in den Karten 2b und 3b lokalisiert. Dabei nimmt i die Werte 1 und 2 an. Der Eventbuilderprozeß EB, der Testprozeß und die Formatierung laufen auf der Karte 1a ab.
Nach dem Hochfahren der Überrahmen (Booten), bearbeitet jeder Controller eine Startliste, in der u. a. verzeichnet ist, welche Programme in die eigene CPU geladen werden sollen. Der Controller holt sich diese Programme über sein lokales Dateisystem (NFS (Network File System), OS-9net). Lediglich der Programmablauf wird später von dem angeschlossenen Subeventreader gesteuert.
Nach dem aktivieren der C40-er in der ISA-Box und des Experiment-PCs wird dann für jeden EA{i}{j}-Prozeß ein Konfigurationsdatensatz zur Parametrierung des gewählten Programms in den zugeordneten Controller geladen. Dies geschieht via Ethernet.
Die Prozesse für die Karten 1b bis 3b werden durch C40-Programme beschrieben. Diese können leicht über eine Interprozessorverbindung (Link), von einem Entwicklungssystem aus (Karte 0), geladen werden. Nach Abschluß der Entwicklungsarbeiten können die Programme über den PCI-Bus des Hauptrechners geladen werden. Dazu müssen passende Treiber geschrieben werden (vgl. 6.4.4).
Wenn alle Programme ablauffähig sind, kann die Datenerfassung gestartet werden. Einige Events können zur Überwachung, während der Datenerfassung, auf dem Experiment-PC dargestellt werden. Dieser Rechner kann auch später die auf Magnetband gesammelten Daten verarbeiten und visualisieren.

6.5. Zusammenfassung

In diesem Kapitel wird eine optimierte Struktur der Datenerfassungssysteme für COSY-Jülich dargestellt. Dadurch, daß eine Kompression der erfaßten Daten parallel zur Datenübertragung möglich wird, ist es zu erwarten, daß Datenerfassungsraten von bis zu 6,25 Mbyte/s erreicht werden können. Das entspricht einer Verbesserung um den Faktor 5.
Das Konzept für die Implementierung der optimierten Hardware wurde vorgestellt.

In der nun folgenden Gesamtzusammenfassung wird noch eine mögliche Verbesserung dieser Realisierung angeben.

7. Zusammenfassung

Diese Arbeit zeigt, wie ein konventionelles Datenerfassungssystem durch die Anwendung des Prinzips der parallel ablaufenden und kommunizierenden sequentiellen Prozesse (CSP) optimiert werden kann. In Kapitel 1.5 sind dazu folgende Forderungen gestellt worden:

1. Höhere Datenerfassungsrate (> 1,25 Mbyte/s)
2. Optimiertes Softwarekonzept
3. Fähigkeit zur On-Line-Kompression

Um den Entwicklungsaufwand für neue Hardware gering zu halten, wurde eine Literaturrecherche durchgeführt, in der bisher gefundene Lösungen ähnlicher Problemstellungen analysiert wurden.
Als Ergebnis wurde eine typische Struktur gefunden, die verifiziert, daß der folgende Ansatz für Datenerfassungssysteme am COSY-Jülich sinnvoll ist.
Die Beschreibung eines Experimentaufbaues wird am Beispiel des Experiments COSY 11 durchgeführt. Über diesen Einstieg in die Problematik wird noch einmal klar gemacht, wie die Struktur eines Datenerfassungssystems in der Hochenergiephysik beschaffen sein muß.
Die theoretische Verallgemeinerung, mit Hilfe der Programmiersprache occam 2 zur Problembeschreibung, läßt jedoch Strukturschwächen erkennen, die dann durch meinen neuen Lösungsansatz behoben werden.
Die neue Struktur liefert einen Algorithmus, der für alle künftigen Streuexperimente am COSY benutzt werden kann. Das folgende Bild zeigt eine Übersicht der möglichen Realisierungen:

Allgemein:	Programmier-sprache	Interprozessor-kommunikation	Echtzeit	Prozessoren
Alt:	C	OS-9net	OS-9	680X0 (CISC)
Neu:	parallel C	parallel C - Routinen	Echtzeitkern	C40 (DSP)
Zukunft 1:	occam 2	Hardware	Hardware	T9000 (Transputer)
Zukunft 2:	occam 2	Hardware	Hardware	"Power Transputer"

Bild 7-1: Übersicht der Datenerfassungssysteme für COSY

Allgemein braucht man ein Multiprozessorsystem mit einer passenden Programmiersprache, dessen Prozessoren möglichst einfach und schnell miteinander kommunizieren können. Das System muß auch echtzeitfähig sein und geeignete Prozessoren verwenden. Bisher löst man diese Aufgabe am COSY mit der

Programmiersprache C, der Kommunikationssoftware OS-9net und dem Echtzeitbetriebssystem OS-9 auf Motorola 680x0 Prozessoren. Der Anteil an komplexen Softwarekomponenten ist also recht hoch und verlangsamt so das gesamte System.
Ideal wäre ein System, in dem nur die Anwenderprogramme in Software kodiert sind und in dem andere Funktionen durch Hardware realisiert werden, also ein System, das z. B. in occam 2 programmierbar ist und mit schnellen Transputern arbeitet (Zukunft 1:).

In dieser Arbeit wurde eine Kompromißlösung vorgestellt, die durch einfachere, und damit auch schnellere, Software-Komponenten und geeignetere Prozessoren (C40 von Texas Instruments) ausgezeichnet ist /TI 91.x/ /TI 92.x/ /TI 93.x/ /3L 92/ /3L 93.x/.
Die Ziele 1. bis 3. sind sicherlich mit diesem System erreichbar. Für eine Realisierung mußte jedoch neue Hardware entwickelt werden. Im Anhang ist insbesondere die PCI-Bus zu C40-Bus Schnittstelle dokumentiert. Dies vermittelt einen Eindruck von der Komplexität der Hardware.

In Zukunft sind zwei verbesserte Lösungen denkbar (Zukunft 1 u. 2:): Falls der T9000 jemals fehlerfrei zu erwerben sein sollte, kann das System durch einen Neuentwurf vereinfacht und dennoch verbessert werden. Es ist aber wahrscheinlicher, daß sich durch den Entwurf eines Moduls mit einem PowerPC Mikroprozessor, einer PCI-Schnittstelle und verbesserten Kommunikationsmöglichkeiten (Spezieller Chip für einen PowerPC für Kommunikationsschnittstellen-Verbindungen) flexiblere und leistungsfähigere Lösungen erzielen lassen. Solch ein System kann logisch wie ein Transputer mit hoher Rechenleistung benutzt werden /PARS 94/.
Das hier erarbeitete Konzept wird sich somit auch auf zukünftige Hardware-Strukturen abbilden lassen.

7.1. Erwartete Leistungsdaten

Wie sieht nun die erwartete Verbesserung durch das optimierte System (Bild 6-10 und Bild 6-11), verglichen mit dem ursprünglichen Lösungsansatz (Bild 4-1) aus?

7.1.1. Maximale Datenerfassungsrate des VMEbus-Systems

Das ursprüngliche System, mit einem VMEbus-Überrahmen als Eventbuilder, hat als Dateneingänge zwei VICbus-Stränge mit bis zu 2 x 33 Mbyte/s Transferrate. Aber schon der VMEbus, mit seiner praktisch erreichbaren Transferrate von 30 Mbyte/s, bremst diese Datenströme. Da die Subeventreader mit Motorola CPUs keine Kommunikationsbandbreite parallel zu Rechenoperationen bereitstellen, ist eine Kompression der Daten kaum möglich. Damit ist die

gesamte Datendurchsatzrate gleich der Datenerfassungsrate und auf 1,25 Mbyte/s begrenzt, falls ein entsprechend schnelles Magnetband verwendet wird /DEC 94/.

7.1.2. Maximale Datenerfassungsrate des C40-Systems

Bei dem optimierten System mit TMS320C40 Signalprozessoren als Subeventreader und Eventbuilder wird die maximale Eingangsdatenrate von 2 x 33 Mbyte/s durch die Kommunikationsschnittstellenverbindungen auf etwa 2 x 13,33 Mbyte/s begrenzt. Diese Werte kommen durch Signalverzögerungen auf den Leitungen zustande /BORG 93/ /LSI 93.3/ /VERM 93/. Die erreichbare Datenrate ist vor dem Eventbuilder mit 26,66 Mbyte/s vergleichbar mit dem VMEbus-System.
Die Bandbreite des DMA-Koprozessors ist zwar auf 40 Mbyte/s begrenzt, es bleibt aber noch genügend Bandbreite, um das Magnetband auszulasten
((40 - (2 x 13,33)) Mbyte/s = 13,33 Mbyte/s >> 1,25 Mbyte/s).
Der entscheidende Vorteil des C40-Systems ist somit die parallele Verfügbarkeit von zusätzlicher Rechenleistung und Kommunikationsbandbreite. Denn: Erreicht man eine mittlere Kompressionsrate von 1 zu 5, so erhöht sich die gesamte Datenerfassungsrate auf 5 x 1,25 Mbyte/s = 6,25 Mbyte/s! Dies scheint angesichts des zu betreibenden Aufwands ein geringer Gewinn, jedoch ist zu bedenken, daß hier durch genaue Kenntnisse der Physik des Experiments noch intelligentere Anpassungen möglich werden, und so die Durchführung zukünftiger Experimente erst realisierbar wird!
Außerdem ist so die On-Line Formatierung der Daten im ZEBRA-Format (vgl. 8.3) möglich, wodurch die spätere Datenauswertung beschleunigt wird, weil die benutzten Softwarepakete dieses Format erwarten.
Die maximale Datenerfassungsrate ist mit 26,66 Mbyte/s um den Faktor 21,33, also um eine Größenordnung, besser als beim VMEbus-System.

8. Anhang

8.1. Abkürzungen

ALT	ALTernativkonstruktor (occam 2)
ASIC	Application Specific Integrated Circuit
C	prozedurale Programmiersprache
C, parallel	C mit Erweiterungen zur Realisation des CSP-Prinzips
CAD	Computer Aided Design
CAMAC	Computer Aided Measurement and Control
CERN	Organisation Européenne pour la Recherche Nucleaire
CES	Creative Electronic Systems
CISC	Complex Instruction Set Computer
COFF	Common Object File Format
COSY	Cooler Synchrotron
CPU	Central Processing Unit
CSP	Communicating Sequential Processes
C40	TMS320C40 (Signalprozessor von TI)
DESY	Deutsches Elektronen-Synchrotron (in Hamburg)
DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
DV	Datenverarbeitung
EA	Event-Acquisition
EB	Eventbuilder
ECL	Emitter Coupled Logic
EISA	Extended Industrial Standard Architecture
FIFO	First In First Out
GeV	Giga (10^9) Elektronenvolt
HW	Hardware
idt	integrated devices technology
IKP	Institut für Kernphysik
ISA	Industrial Standard Architecture
JTAG	Joint Test Action Group

kW	Kilo Worte = 4096 bytes
KS	Kommunikationsschnittstelle (Link)
KSA	Kommunikationsschnittstellenadapter
MAP	Manufacturing Automation Protocol
MMS	Manufacturing Message Specification
MS-DOS	Microsoft-Disk Operating System
MW	Mega Worte = 4096 kbytes
NFS	Network File System
OS	Operating System
PAL	Programmable Array Logic
PAR	PARallelkonstruktor (occam 2)
PAW	Physics Analysis Workstation
PC	Personal Computer
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PLD	Programmable Logic Device
RAM	Random Access Memory
RISC	Reduced Instruction-Set Computer
ROM	Read-Only Memory
SCSI	Small Computer Systems Interface
SEQ	SEQuenzkonstruktor (occam 2)
SER	Subeventreader
SPOX	Signalprocessor Operating System X (UNIX-ähnlich)
SW	Software
TBC	Test Bus Controller
TCP/IP	Transport Control Protocol/Internet Protocol
TI	Texas Instruments
TIM-40	Texas Instruments Module for TMS320C40 Systems
TOF	Time-of-Flight
VDB	VSB Differential Bus
VICbus	VME Inter-Crate Bus
VMEbus	Versa Module Europe Bus
VSB	VME Subsystem Bus
VXI	VME eXtension for Instrumentation

ZM Zustandsmaschinen

8.2. Glossar

D Definition
E Erläuterung

Die Definitionen und Erläuterungen in diesem Glossar wurden teilweise mit Hilfe von /ZWOL 94/ /MUSI 88/ und /ZWOL 93/ formuliert. Begriffe, die durch *Kursivschrift* gekennzeichnet sind, sind auch in diesem Glossar zu finden.

Baryonen

D Baryonen (= "schwere Teilchen" in der ursprünglichen Bedeutung) Ein Baryon ist ein Teilchen, das durch ein Quark-Triplett gebildet wird. *Nukleonen* sind spezielle Baryonen.

Datendurchsatzrate

D Ist die Menge an Daten pro Zeiteinheit, die ständig von einem Gesamtsystem verarbeitet werden kann.

Datenerfassungsrate

D Ist die Menge an Daten pro Zeiteinheit, die ständig in ein Gesamtsystem, an den digitalen Eingängen, fließen kann. Da nicht alle erfaßten Daten das Gesamtsystem durchsetzen, kann die Datenerfassungsrate größer sein als die *Datendurchsatzrate*.

Event

E Ein Event (Ereignis) ist das Abbild eines physikalischen Geschehens eines Experiments, in einem bestimmten Zeitintervall, in Form einer Datei von Meßdaten.

Eventbuilder

E Ein Eventbuilder ist eine HW oder SW, die aus *Teil-Event*-Dateien eine einzige *Event*-Datei (kurz: *Event*) formt.

Hadronen

D Hadronen sind Teilchen, die der starken Wechselwirkung unterliegen, also *Baryonen* oder *Mesonen*.

Instrumentierungssystem

E Ein Instrumentierungssystem ist eine Menge von Modulen (Digitalisierungs-Hardware), die in einem physikalischen oder logischen Zusammenhang stehen.

Leichtgewicht-Prozeß
E *Thread*

Leptonen
D Leptonen (= "leichte Teilchen" in der ursprünglichen Bedeutung). Heute: Teilchen, die nicht der starken Wechselwirkung unterliegen, d. h. die sich normalerweise nicht in einem Atomkern befinden. Leptonen sind die zweite Gruppe der Elementarteilchen (*Quarks*). Leptonen sind: Elektronen, Neutrinos (Ungeladenes Elektron = Elektronenneutrino), Myonen, Myonenneutrino, Tauon und Tauonneutrino (letzteres ist bis heute noch nicht nachgewiesen worden).

Mesonen
D Mesonen (= "mittelschwere Teilchen" in der ursprünglichen Bedeutung). Ein Meson ist ein Teilchen, das durch ein *Quark*-Antiquark Paar gebildet wird. Es ist instabil und entsteht nach Kollisionen oder Zerfällen.

Nukleonen
D Nukleonen (= Kernteilchen) Nukleonen sind Protonen oder Neutronen. Nukleonen sind eine kleine Teilmenge der *Baryonen*.

Prozeß
D *Task*

Quarks
D Quarks sind die erste Gruppe der Elementarteilchen (*Leptonen*). Sie gelten derzeit als unteilbar, nicht isolierbar und treten als Tripletts (*Baryonen)* oder *Quark*-Antiquark Paare (*Mesonen*) auf.

Subevent
E Ein Subevent ist eine Datenmenge, die in einem bestimmten Zeitinterval aus einer Menge von Digitalisierungsmodulen ausgelesen wird.

Subeventreader
E Ein Subeventreader liest die *Subevents* von einem VICbus-Strang und bildet daraus eine *Teil-Event*-Datei.

Task

D "Eine Task ist eine in sich geschlossene Aufgabe, repräsentiert durch einen Programmteil, ein vollständiges Programm oder auch eine Programmfolge" /DIN 87/, mit eigenem Stapel- und Datenspeicherbereich (Stack and Data).

Teil-Event

E Ein Teil-Event ist eine Datei, die aus einer Sequenz von *Subevents*, ausgelesen von einem VICbus-Strang, und aufgesetzten Schlüsselinformationen (Header) besteht.

Thread

D Ein Thread (Ausführungsfaden, Leichtgewicht-Prozeß) ist ein Prozeß (*Task*) mit eigenem Stapelspeicherbereich (Stack). Den Datenspeicherbereich (Data) muß sich ein Thread u. U. mit einem oder mehreren anderen Threads teilen.

8.3. Das ZEBRA-Format

Das "ZEBRA Exchange Data Structure Format" beschreibt Datenstrukturen für den Datenaustausch.
Dieses Datenformat hat in der Hochenergiephysik eine weite Verbreitung gefunden /ZOLL 91/.

Die verschiedenen Files auf dem Magnetband sind durch Markierungen voneinander getrennt. Ein File im ZEBRA-Format ist in mehrere physikalische Datenblöcke (Records) (phrs) konstanter Länge aufgeteilt. Die Länge ist ein ganzzahliges Vielfaches von 30 Worten. Ein physikalischer Record besteht aus einigen logischen Records (lors) die unterschiedliche Typen haben können. In einem logischen Record vom Typ 2 der Daten vom Typ 3 beinhaltet, sind dann im allgemeinen mehrere Events untergebracht.
Die Speicherung der Events erfordert also die hierarchische Verschachtelung von mit Dateischlüsseln (Headers) versehenen Datenblöcken.

8.3.1. Typen der Logischen Records

Es gibt sechs verschiedene Typen von logischen Records:

Typ 1:	Start oder Ende eines Durchlaufs (Beginn oder Ende eines Files)
Typ 2:	Datenstruktur: "Event-Daten"
Typ 3:	Datenstruktur: "Event-Daten, Fortsetzung"
Typ 4:	Datenstruktur: "Fortsetzung"
Typ 5 oder 6:	Füll-Records (Zum Auffüllen von physikalischen Records)

8.3.2. Typen der Daten

Es gibt acht verschiedene Datentypen in einem logischen Record vom Typ 2 oder 3:

Typ 1:	Hardware-Konfigurationsdaten
Typ 2:	Hardware-Initialisierung
Typ 3:	Events
Typ 4:	COSY-Kontrolle
Typ 5:	Kontrolldaten
Typ 6:	Zeitmarken (Time Stamps)
Typ 7:	Analyse-Konfiguration
Typ 8:	On-line Analyse

8.3.3. Die ZEBRA-Dateistruktur

Das folgende Schema gibt eine Vorstellung vom Aufbau einer ZEBRA-Datei:

```
--*****************************************************************
-- 1. phr:     phrhdr (Bit #20000000 = 1: Anfang des Files)
--             1.lor Typ 1 (Anfang oder Ende des Files) nrun=1
--             2.lor Typ 2 (Daten), Datentyp 6 (Zeitmarke)
--             3.lor Typ 2 (Daten), Datentyp 1 (Hardware-Konfiguration)
--             4.lor Typ 2 (Daten), Datentyp 2 (Hardware-Initialisierung)
--             ...
--             n.lor Typ 5 (Füllung (padding))
--*****************************************************************
-- 2. phr:     phrhdr (Bit #E0000000 = 0)
--             1.lor Typ 2 (Daten), Datentyp 3 (Event), nevents=1
--             2.lor dito.
--             ...
--             n.lor Typ 5 (Füllung (padding))
--*****************************************************************
-- i. phr      dito.
--*****************************************************************
-- m. phr:     phrhdr        (Bit #40000000 = 1) Ende des Files
--letzter phr: 1.lor Typ 2 (Daten), Datentyp 3 (Event), nevents=1
--             2.lor dito.
--             ...
--             n-2.lor Typ 2 (Daten), Datentyp 6 (Zeitmarke)
--             n-1.lor Typ 1 (Anfang oder Ende des Files) nrun=0
--             n.lor Typ 5 (Füllung (padding))
--*****************************************************************
```

Nach dem physikalischen Register-Schlüssel (Physical Record Header (phrhdr)), der auch Steuerungsblock genannt wird, folgen Informationen über den Zeitpunkt, die Konfiguration und die Parameter des Experiments. Erst ab dem zweiten phr (physical record) findet man gemessene Daten. Jeder phr ist mit einem Füll-Record abgeschlossen. Im letzten phr steht vor der "Ende des Files" Markierung noch einmal eine Zeitinformation.

8.4. Die C40-Bus zu S_PCI-Bus Schnittstelle

In Bild 6-5 und Bild 6-7 ist das Blockschaltbild der C40-Karte zu sehen. Die komplexe C40-Bus zu S_PCI-Bus (Sekundärer PCI-Bus) Schnittstelle wird hier näher erläutert. Wie in Kapitel 6.4.1 bereits erwähnt, konnte diese Schnittstelle nicht mit Hilfe eines ASIC realisiert werden, weil im Zentrallabor für Elektronik des Forschungszentrums Jülich die notwendigen Werkzeuge fehlen. Daher wählte ich die programmierbaren PLDs (Programmable Logic Devices) MACH 435 der Firma AMD (Advanced Micro Devices), weil diese Bausteine die schnellsten und komplexesten sind, die im Zentrallabor für Elektronik benutzt werden können. Diese Bausteine erlauben es Zustandsmaschinen mit einer Taktrate von bis zu 40 MHz zu realisieren. Ein MACH 435 enthält acht fiktive 33V16 PALs (entspricht etwa der Komplexität von elf 22V10 PALs), die über eine Schaltmatrix miteinander verbunden werden können. Der Baustein hat 84 Anschlüsse und enthält 128 Flip-Flops, d. h. es können maximal 128 bit für Registerfunktionen genutzt werden /AMD 93/.
Für die C40-Bus zu S_PCI-Bus Schnittstelle müssen fünf MACH 435 Bausteine eingesetzt werden. Je ein PLD enthält die Master- bzw. Slave-Schnittstelle. Zwei PLDs enthalten die für ein PCI-Gerät notwendigen Konfigurationsregister, dabei enthält ein Baustein die oberen und der andere die unteren 16 bit Worte. Der fünfte Baustein enthält den Arbiter und einen Blockzähler.
Diese Lösung benötigt sehr viel Raum, da ein MACH 435 mit Sockel ca. 3,5 cm x 3,5 cm Platz einnimmt. Beim Design des Layouts für die C40-Karte wurde dies zu einem Problem.

8.4.1. Die C40-Bus zu S_PCI-Bus Master-Schnittstelle

Bild 8-1 zeigt ein Blockschaltbild auf Zustandsmaschinen- bzw. Registerübertragungsebene des MACH-Bausteins, der die Schnittstelle vom Prozessorbus zum S_PCI-Bus, mit der Funktion als PCI-Initiator (Master), realisiert. Die Buszyklussteuerungen sind durch die Verwendung je einer eigenen ZM (Zustandsmaschine) für den C40-Bus bzw. den PCI-Bus entkoppelt. Das Bindeglied zwischen beiden ist die ZM für Anschluß B (Port B) des bidirektionalen FIFOs (BI-FIFO).
Die beiden Seiten der Schnittstelle arbeiten mit unterschiedlichen Taktraten. Der S_PCI-Bus mit 33 MHz (S_clk) und der C40-Bus mit 20 MHz (CH1_clk). Daher ist für einige Zustände eine Doppelsynchronisation der Signale erforderlich, bevor diese abgetastet werden. Dies dient der Vermeidung von metastabilen Zuständen /AMD 90/. In Bild 8-1 ist auch zu sehen, wie die verschiedenen Takte den ZMs und dem BI-FIFO zugeordnet sind. Dabei ist zu beachten, daß die Flaggen mit den Indizes XXab_, also z. B. EFab_ (FIFO mit Richtung a -> b ist leer), synchron zu S_clk sind. Entsprechend sind die Flaggen mit den Indizes XXba_, also z. B. EFba_ synchron zu CH1_clk.

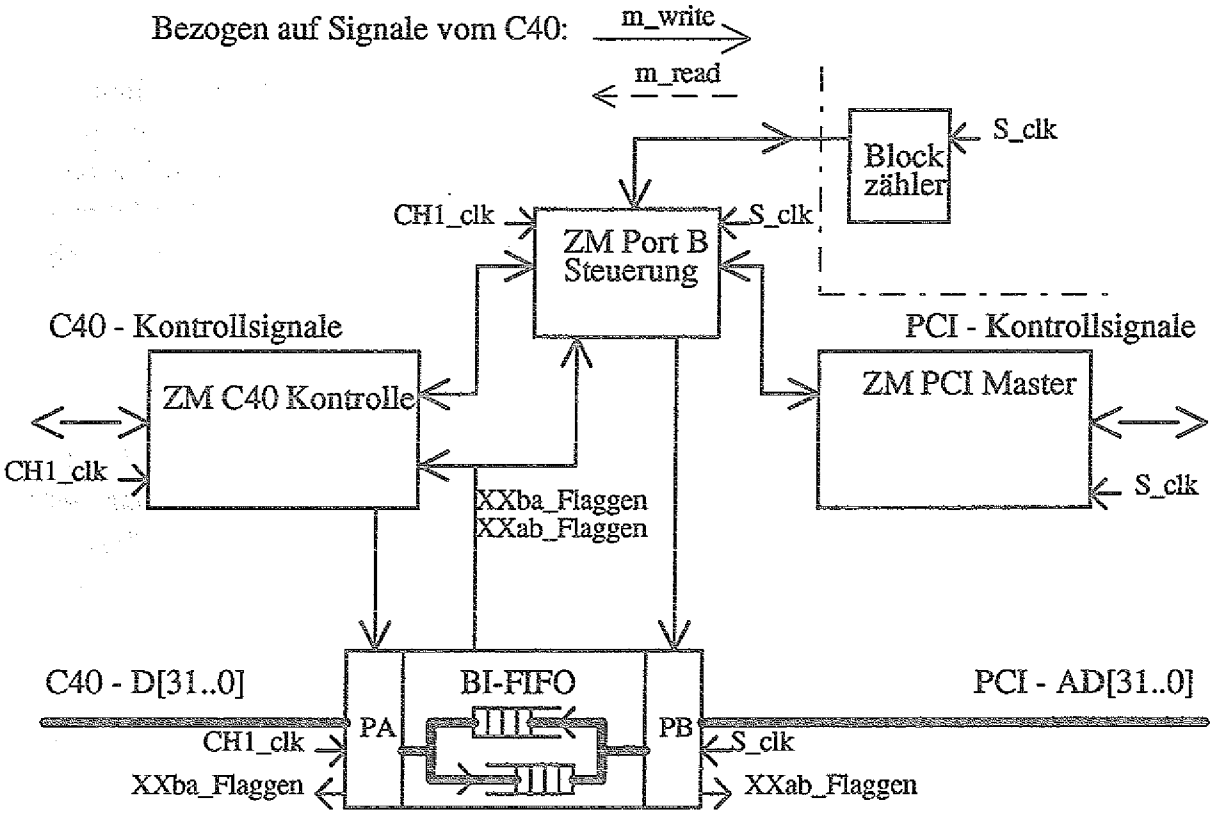


Bild 8-1: C40-Bus zu S_PCI-Bus Master-Schnittstelle

8.4.2. Die C40-Bus zu S_PCI-Bus Slave-Schnittstelle

In Bild 8-2 ist die C40-Bus zu S_PCI-Bus Slave-Schnittstelle (Slave = Target) dargestellt. Für die Takte gilt analoges wie in Kapitel 8.4.1 geschrieben. Die Transferrichtung wird vom jeweiligen Master am S_PCI-Bus festgelegt. Ein Transfer: C40-Bus => S_PCI-Bus wird mit s_read (Slave-Read) bezeichnet. Um diese Schnittstelle zu realisieren, muß man in einen MACH 435 Baustein fünf ZMs durch ein Programm abbilden. Die ZMs zur Erzeugung oder Steuerung der Buszyklen, sowie die Steuerung für die BI-FIFO Anschlüsse (Ports), sind logisch getrennt und nur durch Start-Stopsignale miteinander verknüpft. Diese modulare Vorgehensweise sollte, besonders in der Testphase, die schnellere Beseitigung von Fehlern erlauben.

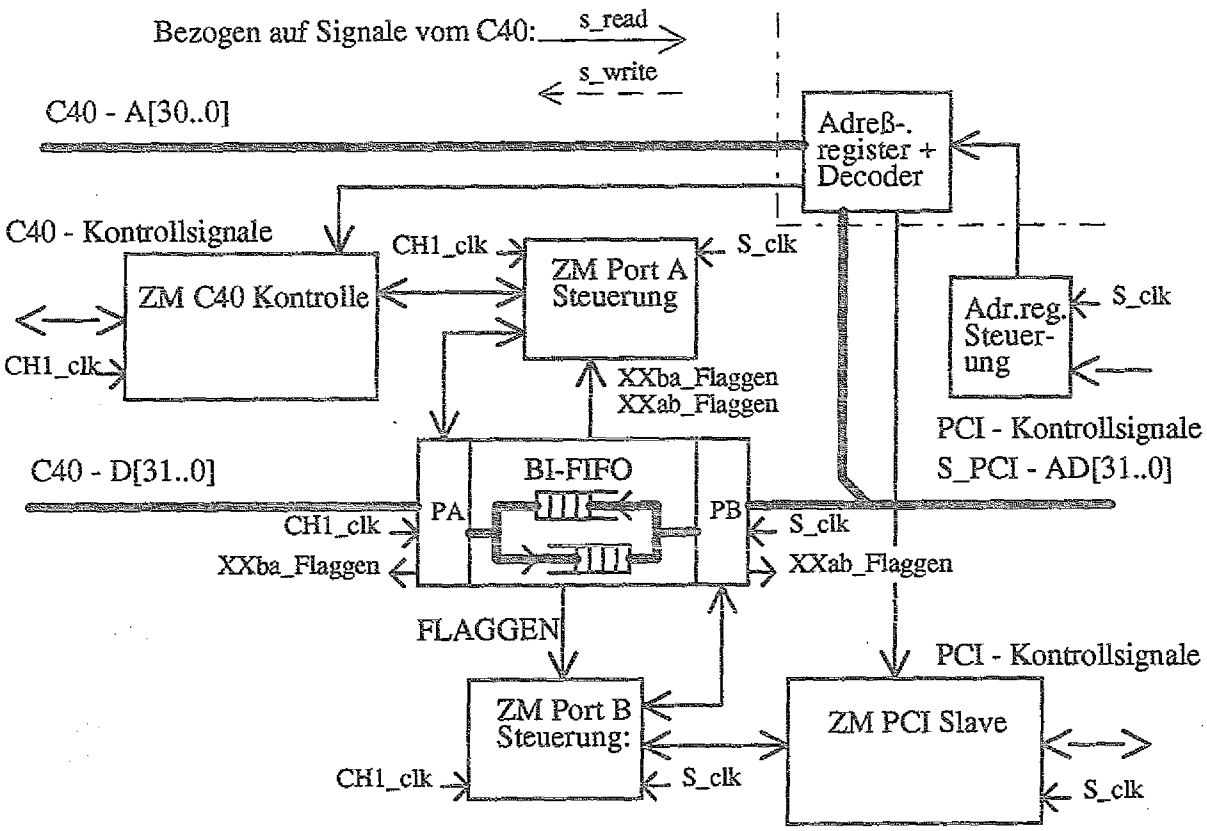


Bild 8-2: C40-Bus zu S_PCI-Bus Slave-Schnittstelle

Der S_PCI-Bus ist mit dem FIFO Port B über einen internen Datenbus B verbunden (vgl. Bild 6-7). Der globale C40-Datenbus (interner Datenbus A) ist mit dem FIFO-Port A verbunden. Eine Datenübertragung zwischen C40 und S_PCI ist also nur über das FIFO möglich. Dadurch werden Einzeltransfers verlangsamt. Der zusätzliche schaltungstechnische Aufwand für einen weiteren direkten Datenpfad war aber nicht vertretbar.
Der S_PCI-Bus "sieht" den Konfigurationsadreßraum und den Speicheradreßraum, bestehend aus C40-Speicher und FIFO Port B, via Einzeltransfers über (auf) das FIFO. Es sind Schreib-Blocktransfers auf FIFO Port B möglich. Dadurch wird, falls nicht maskiert wurde, ein Interrupt für den C40 ausgelöst. Der C40 muß dann dafür sorgen, daß die Daten im FIFO richtig bearbeitet werden.
Falls der C40 das FIFO mit Daten gefüllt hat, sind auch Lese-Blocktransfers möglich.
Je nach Typ des Zugriffes sind nicht immer alle ZMs aktiv. Ein Blocktransfer ist nur beim Zugriff auf das FIFO-Port B Datenregister möglich. Ansonsten führt der Versuch eines Masters zu einem "target-abort" Zyklus. Ist das FIFO nicht bereit, oder greift der Prozessor gerade auf seinen globalen Bus (FIFO Port A) zu, wird ein "target-retry" Zyklus erzeugt. Die Mechanismen "target-abort" und "target-retry" sind in der PCI-Spezifikation beschrieben /INTE 93/.

8.4.3. Die Adreßräume des C40- und des PCI-Busses

8.4.3.1. Mode EB: Verwendung als Eventbuilder

In dieser Betriebsart befindet sich die C40-PCI Schnittstelle, die Speicher-PCI Schnittstelle und die SCSI-PCI Schnittstelle am sekundären Bus der PCI/PCI Brücke (vgl. Bild 6-5). D. h. der C40 ist ein Gerät an diesem Bus. Als Arbiter wird der interne Arbiter in der Brücke verwendet. Hier werden auch die IDSEL-Signale zur Konfigurierung erzeugt. Der Bustakt: S_clk wird durch den Takt P_clk des PCI-Hauptrechners über einen Takttreiber generiert. Die zentralen Ressourcen der C40-PCI Schnittstelle sind jetzt abgeschaltet. (Signal: #C40_CRSC = 1).

8.4.3.2. Mode SER: Verwendung als Subeventreader

In dieser Betriebsart werden die C40-PCI Schnittstelle und eine VICbus-PCI Schnittstelle durch einen **lokalen** PCI-Bus verbunden (vgl. Bild 6-5). D. h. die VICbus-PCI Schnittstelle ist ein Gerät an diesem Bus. Als Arbiter wird der Arbiter in der C40-PCI Schnittstelle verwendet. Hier werden auch die IDSEL-Signale bei der Konfigurierung erzeugt. Der Bustakt S_clk wird durch einen eigenen 33 MHz Oszillator über einen Takttreiber generiert. Die zentralen Ressourcen der C40-PCI Schnittstelle sind jetzt eingeschaltet (Signal: #C40_CRSC = 0).

8.4.3.3. Der Adreßraum des C40, gesehen vom C40

Ein Programm auf dem C40 sieht den Adreßraum wie in Bild 8-3. Es ist zu beachten, das die mit EB (SER) bezeichneten Adreßbereiche nur im Mode EB (SER) vorhanden sind.
Der PCI-Konfigurationsadreßraum und der Speicheradreßraum des C40 überlappen sich. D. h. erzeugt der C40 Buszyklen auf bestimmte Adressen, so werden PCI-Konfigurationszyklen erzeugt.

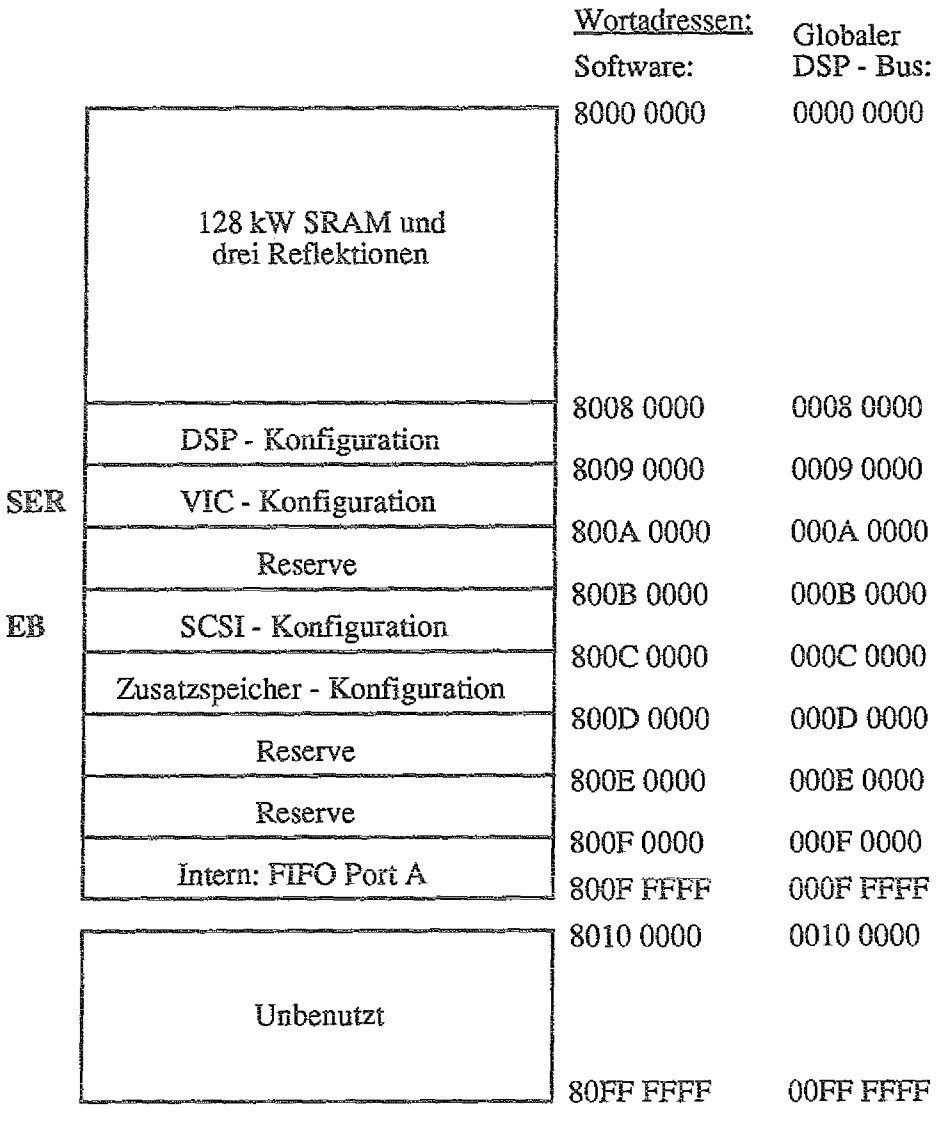


Bild 8-3: Der Adreßraum des C40, vom C40 aus gesehen, Teil 1

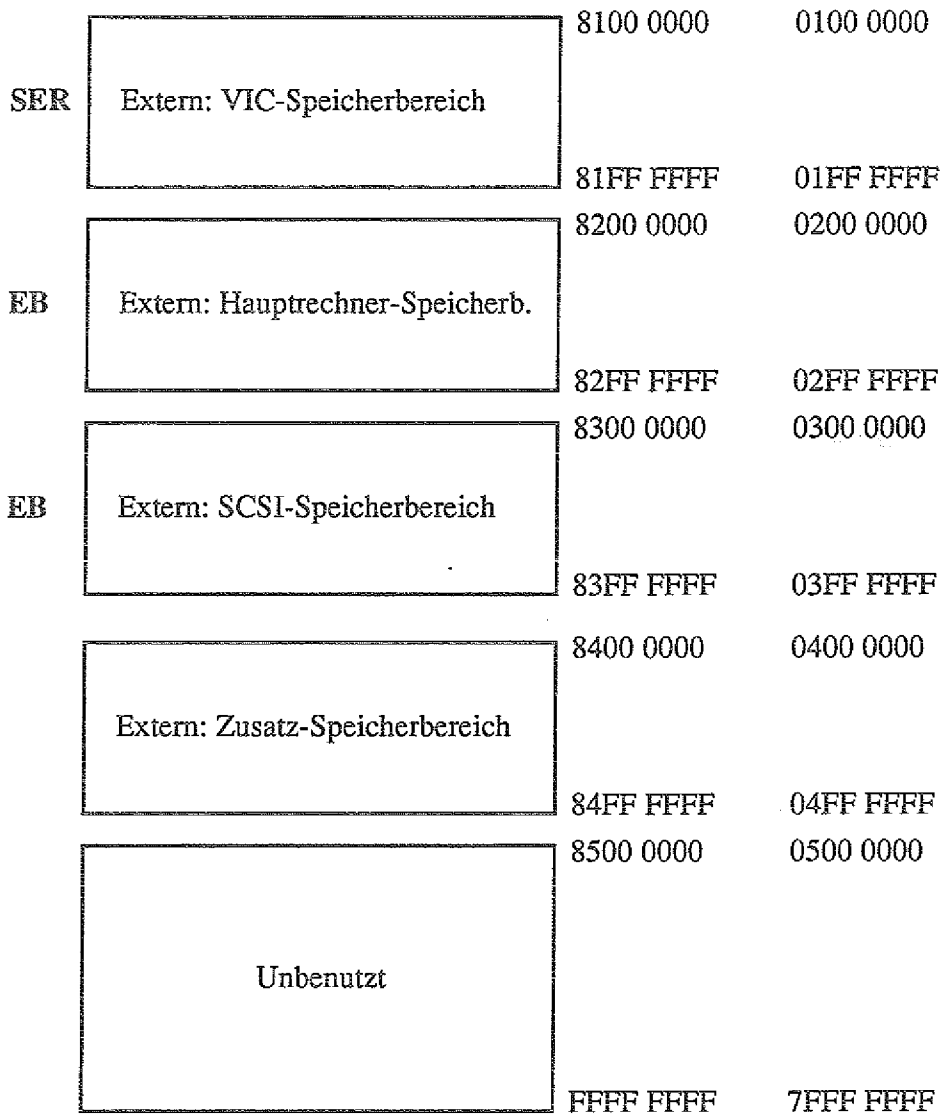


Bild 8-3: Der Adreßraum des C40, vom C40 aus gesehen, Teil 2

8.4.3.4. Der Speicheradreßraum des PCI - Busses

Ein Programm auf dem PCI-Hauptrechner sieht den Speicheradreßraum wie in Bild 8-4.

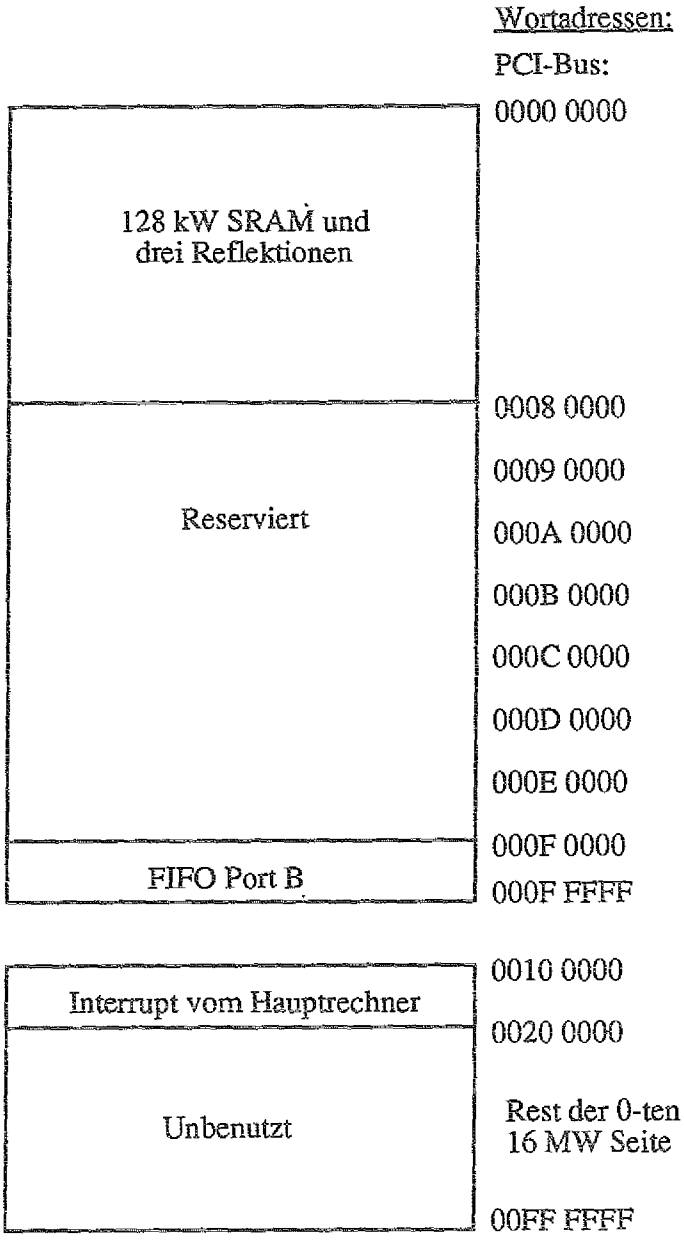


Bild 8-4: Der Speicheradreßraum des PCI-Busses, Teil 1

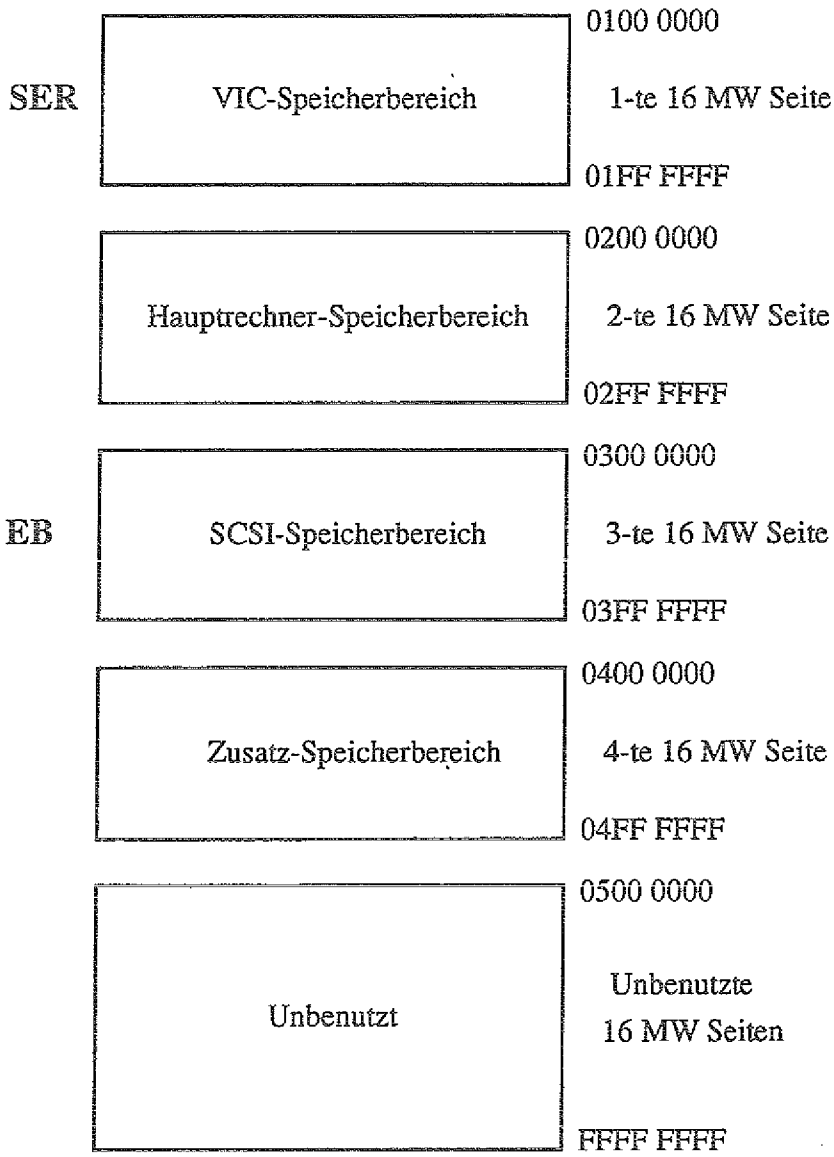


Bild 8-4: Der Speicheradreßraum des PCI-Busses, Teil 2

8.4.3.5. Der Konfigurationsadreßraum des PCI - Busses

Ein Programm auf dem PCI-Hauptrechner sieht den Konfigurationsadreßraum wie in Bild 8-5.

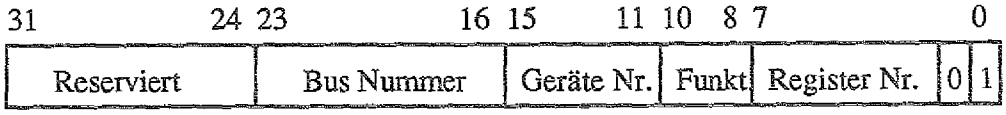


Bild 8-5: Der Konfigurationsadreßraum des P_PCI-Busses

D. h. wird auf dem P_PCI-Bus der Bridge ein Konfigurationszyklus mit P_AD<1:0> = 01 erzeugt, so wird das Adressierungsformat nach Bild 8-5 wirksam. Hat man die richtige Bus Nummer gewählt, so wird auf dem S_PCI-Bus der Bridge ein Konfigurationszyklus erzeugt. Die Geräte Nummer wird dabei auf die S_AD<18:11>-Signale zur IDSEL-Generierung abgebildet. Die Zuordnung erfolgt wie in Bild 8-6.

Geräte Nummer:		S_Bus IDSEL:		Gerät:
P_AD<15:11>	Hex.	S_AD<31:11>	Hex.	
00000	0000 0000h	0000 0000 0000 0000 0000 1	0000 0800h	#C40_IDSEL
00001	0000 0800h	0000 0000 0000 0000 0001 0	0000 1000h	#VIC_IDSEL
00010	0000 1000h	0000 0000 0000 0000 0010 0	0000 2000h	#SCSI_IDSEL
00011	0000 1800h	0000 0000 0000 0000 0100 0	0000 4000h	#MEM_IDSEL
00100	0000 2000h	0000 0000 0000 0000 1000 0	0000 8000h	Unbenutzt
00101	0000 2800h	0000 0000 0000 0001 0000 0	0001 0000h	Unbenutzt
00110	0000 3000h	0000 0000 0000 0010 0000 0	0002 0000h	Unbenutzt
00111	0000 3800h	0000 0000 0000 0100 0000 0	0004 0000h	Unbenutzt
01000-11110	0000 4000h- 0000 F000h	Transaktion abgebrochen		
11111	0000 F800h	Spez. Zykl: R.No.= 00h #S_Rst=0: R.No.= 01h Trans.abge.: R.No.> 01h		

Bild 8-6: Zuordnung der Gerätenummern zu den S_IDSEL-Signalen /DEC 93/

8.4.3.6. Der Konfigurationsadreßraum der C40-PCI Schnittstelle

Die ersten 64 byte des Konfigurationsadreßraums sind wie im PCI-Standard festgelegt implementiert /INTE 93/. Dann folgen 16 byte mit Konfigurationsinformationen, die Institutsweit einheitlich sein sollten. Erst dann folgt das, nur für diese Schnittstelle spezifizierte, C40 Status- und Kommandoregister .

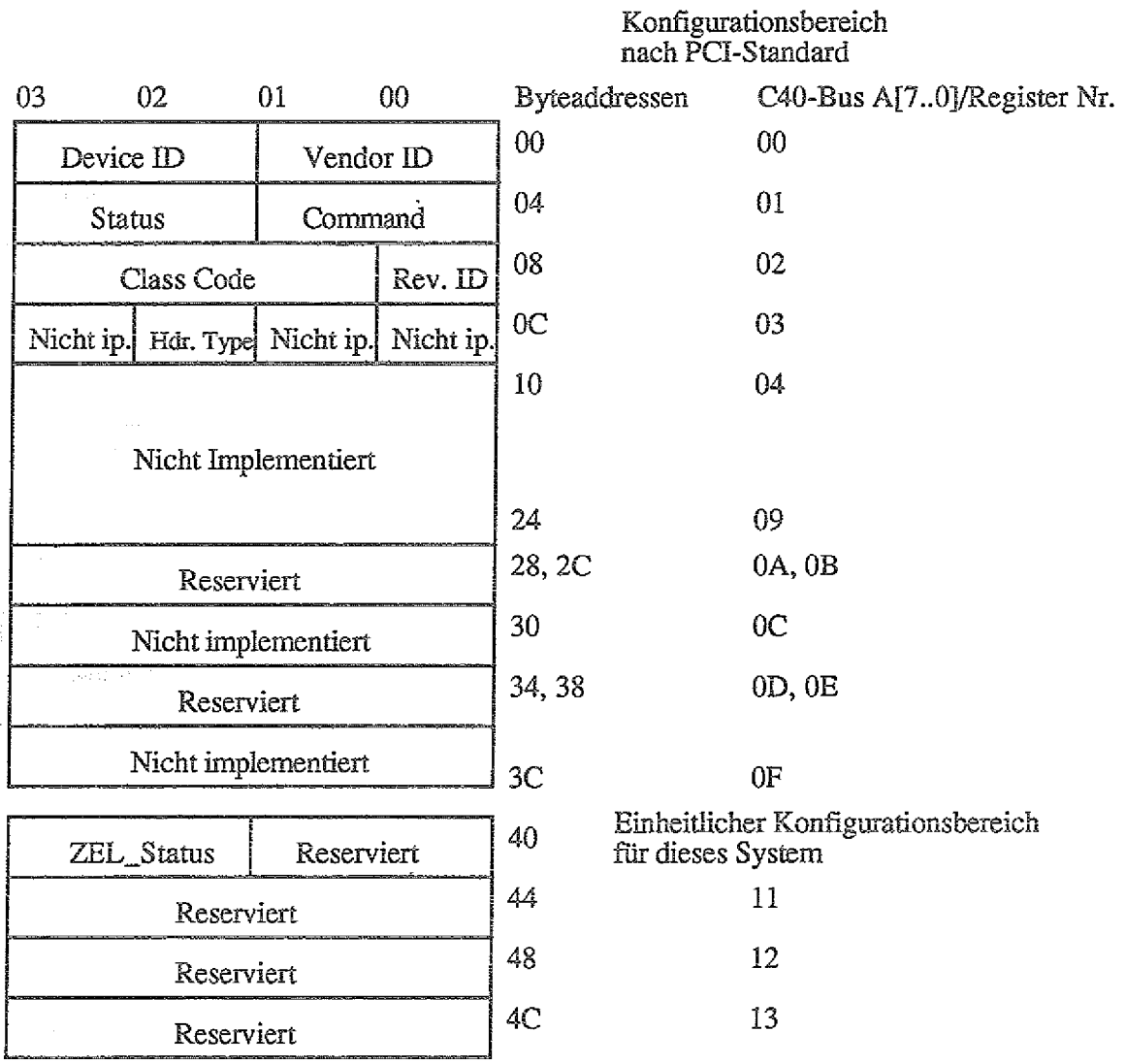


Bild 8-7: Der Konfigurationsadreßraum der C40-PCI Schnittstelle, Teil 1

Spezifischer Konfigurationsbereich
für diese Schnittstelle

		Byteadressen	C-40 Bus A[7..0]/Register Nr.
C40_Status	C40_Command	50	14
C40_INT_FLAGS	Reserviert	54	15
C40_INT_MASK	C40_INT_OUT	58	16
Nicht implementiert	C40_BLK_LEN	5C	17
Nicht implementiert	C40_COUNTER	60	18
Nicht implementiert		64	19
Nicht implementiert		68	1A
Nicht implementiert		6C	1B
Read B->A or Write A->B		70	1C
Bypass - Mode		74	1D
Nicht implementiert		78	1E
Nicht implementiert		7C	1F
A->B FIFO Almost-Empty Flag Offset		80	20
A->B FIFO Almost-Full Flag Offset		84	21
B->A FIFO Almost-Empty Flag Offset		88	22
B->A FIFO Almost-Full Flag Offset		8C	23
Nicht implementiert		90	24
		FC	3F

Bild 8-7: Der Konfigurationsadreßraum der C40-PCI Schnittstelle, Teil 2

8.4.3.7. Die Register im Konfigurationsadreßraum

Die hier dokumentierten Register legen das Verhalten der C40-Bus - PCI-Bus Schnittstelle fest. Dabei werden folgende Schreibweisen verwendet: Eine 0 als Platzhalter in einer Bitposition sagt aus, daß dieses Bit nicht implementiert ist, d. h. es wird als Null gelesen und ist nicht veränderbar. Ein RO-Bit (Read-Only Bit) ist nur lesbar und macht den aktuellen Zustand einer Signalleitung erkennbar. Ein CB (Control Bit) ist schreib- und lesbar und legt den Zustand einer Signalleitung in der Hardware fest. Ein SB (Status-Bit) liefert beim Lesen eine Eins, wenn die Setzbedingung für dieses Bit vorher erfüllt wurde. Beim Schreiben einer Null auf ein SB wird es nicht verändert, beim Schreiben einer Eins wird es gelöscht. Dies entspricht der Definition für Status-Bits im PCI-Standard /INTE 93/. Dadurch ist das selektive Rücksetzen mehrerer Bits mit einer Schreiboperation möglich.

8.4.3.7.1. Das PCI-Statusregister: PCI_Status

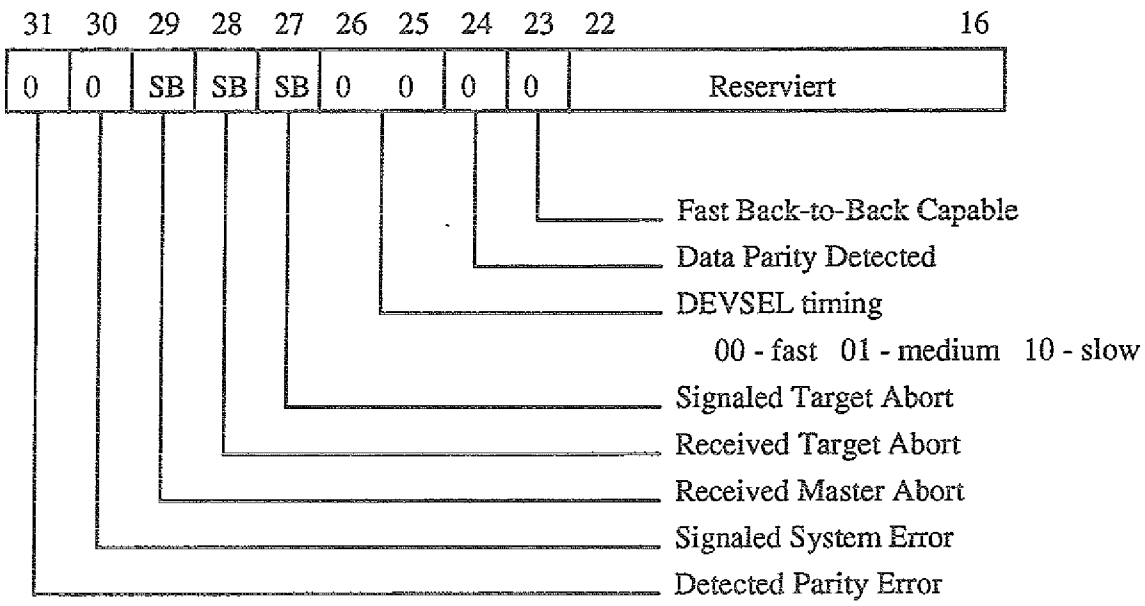


Bild 8-8: Das PCI-Statusregister: PCI_Status

Die genaue Funktion der einzelnen Bits ist im PCI-Standard dokumentiert.

8.4.3.7.2. Das PCI-Kommandoregister: PCI_Command

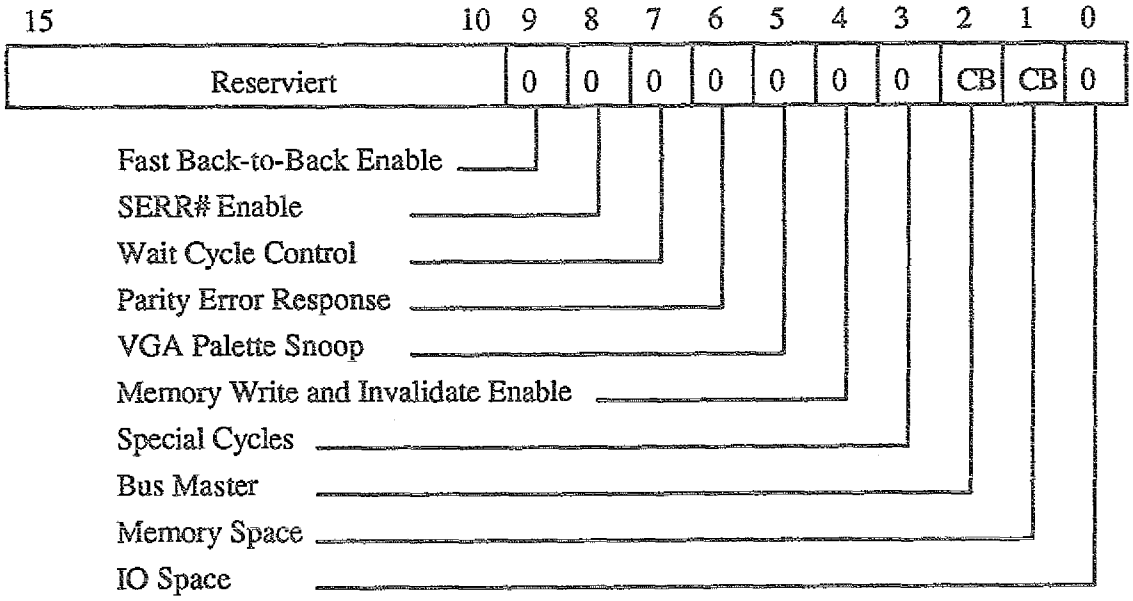


Bild 8-9: Das PCI-Kommandoregister: PCI_Command

Die Funktion der einzelnen Bits ist im PCI-Standard dokumentiert.

8.4.3.7.3. Das erweiterte PCI-Statusregister: ZEL_Status

Um einer CPU mehr Informationen über den Status nach einem PCI-Zyklus zu geben, sind im Register ZEL_Status zwei weitere Statusbits implementiert. Diese heißen, anlog zu "Target abort received" und "Target abort signalled", "Target retry received" und "Target retry signalled". Die Positionen der Bits sind Bild 8-10 zu entnehmen.

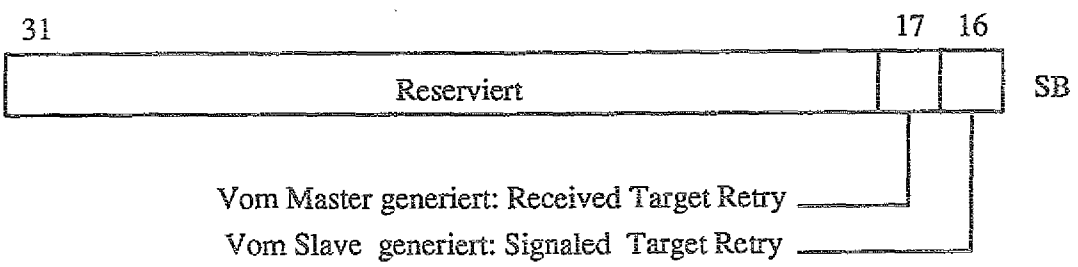


Bild 8-10: Bitpositionen im ZEL_Statusregister.

Die Bits zeigen an, ob der jeweilige Zyklus aufgetreten ist.

8.4.3.8. Der schnittstellenspezifische Konfigurationsadreßraum

8.4.3.8.1. Das C40-Statusregister: C40_Status

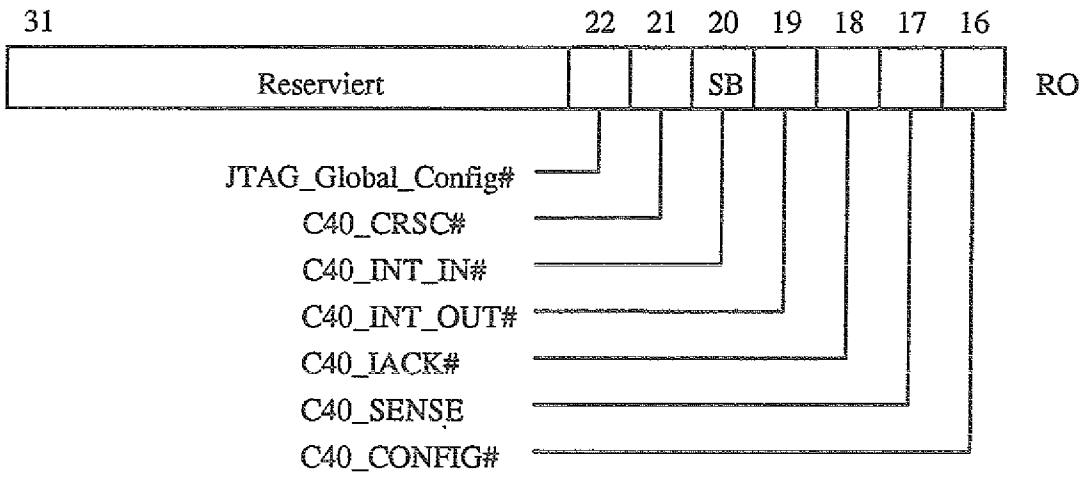


Bild 8-11: Bitpositionen im C40-Statusregister.

Die Bedeutung der Bits im aktiven Zustand ist:

JTAG_Global_Config#: Die globale(!) Konfigurationsleitung der mit dem JTAG-Bus verketteten Karten kann hier abgefragt werden. Ist dieses Bit = 1, dann sind alle vernetzten Prozessoren konfiguriert, d. h. "gebootet"

C40_CRSC#: Die Funktionen der "Centralen ReSsourCe" sind eingeschaltet. Diese sind: Arbitrierung, xx_IDSEL-Signal Generierung, Taktversorgung.

C40_INT_IN#: Der C40 erhält einen Interrupt.

C40_INT_OUT#: Der C40 sendet einen Interrupt.

C40_IACK#: Der C40 aktiviert seinen IACK# Kontaktstift.

C40_SENSE: Es ist ein TIM-40 Modul vorhanden.

C40_CONFIG#: Z. Z. erfolgt die Konfigurierung des C40, d. h. die Boot-Routine ist noch aktiv.

8.4.3.8.2. Das C40-Kommandoregister: C40_Command

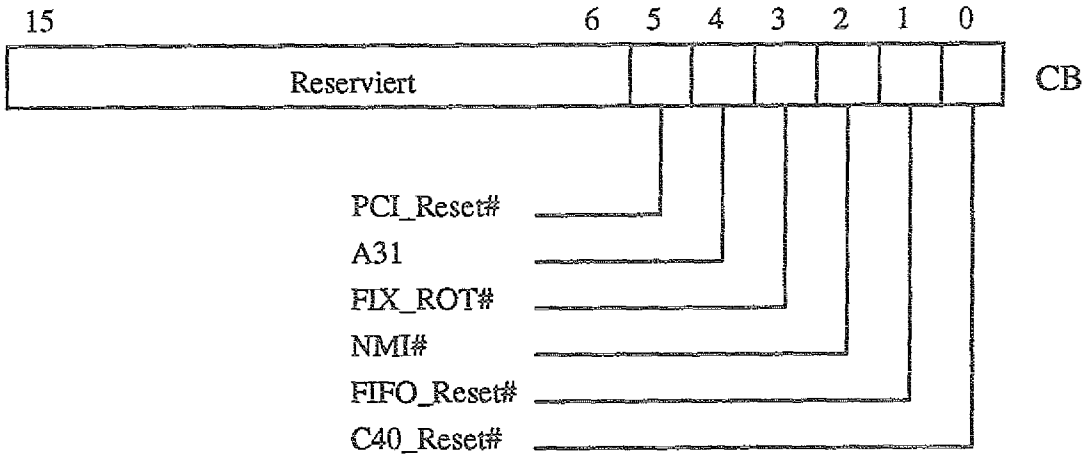


Bild 8-12: Bitpositionen im C40-Kommandoregister.

Die Bedeutung der Bits im aktiven Zustand ist:

PCI_Reset#:	Die RES# Leitung des PCI-Busses wird aktiviert.
A31:	Hier kann der Wert des Adreßbits A31, für Master-Operationen des C40, eingestellt werden. Bei Zugriffen auf den C40 als Slave ist dieses Bit ohne Wirkung. Andere Master sollten A31 = 0 setzten.
FIX_ROT#:	Festlegung des Arbitrierungsmodus, z. Z. nicht implementiert.
NMI#:	Wert der nicht maskierbaren Interrupt-Leitung zum C40.
FIFO_Reset#:	Rücksetzen aller FIFO-Parameter.
C40_Reset#:	Rücksetzen des C40 und der Kommunikationsport-Puffer-Logik.

8.4.3.8.3. Das C40-Interrupt-Flaggenregister: C40_INT_FLAGS

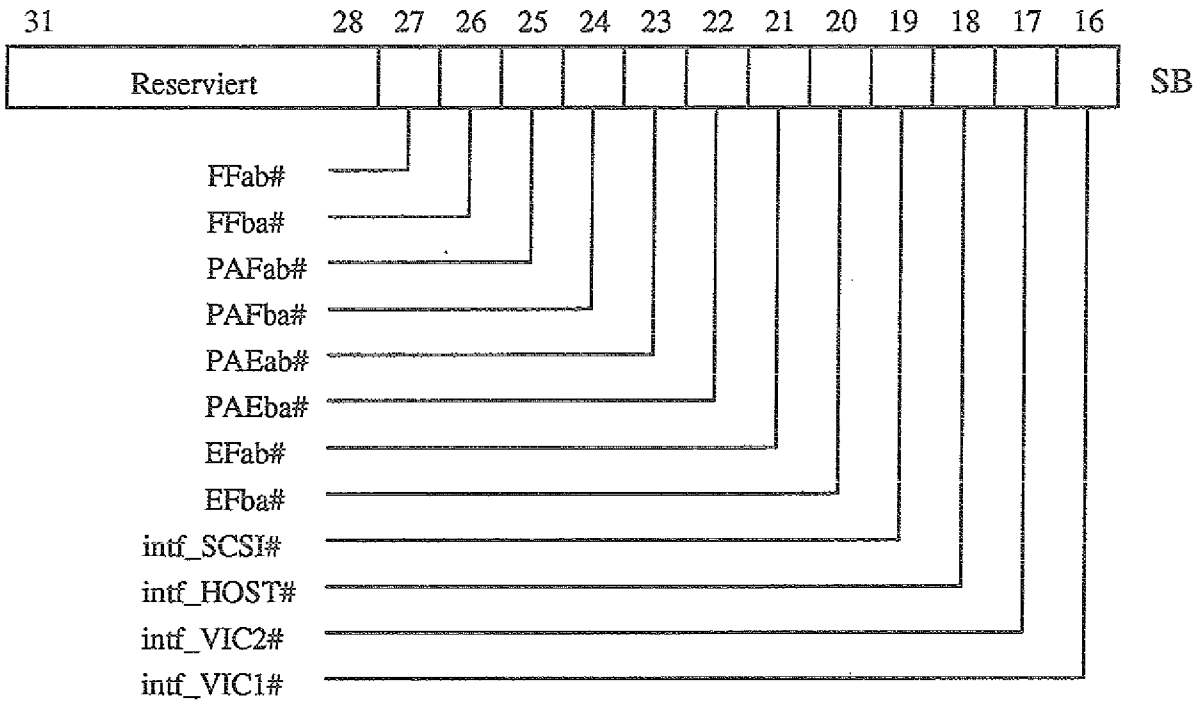


Bild 8-13: Bitpositionen im C40-Interrupt-Flaggenregister.

Ein gesetztes Bit im C40-Interrupt-Flaggenregister erzeugt das Signal C40_INT_IN für den C40 und löst damit einen Interrupt aus. Es sei denn, das entsprechende Bit ist im Maskenregister maskiert, d. h. gesetzt. Die Interrupt-Behandlungsroutine liest dann diese Register, um die Information, welcher Interrupt ausgelöst wurde, zu gewinnen. Erst dann kann der Interrupt bearbeitet werden.
Dabei bedeutet intf_XX = Interrupt von der Quelle XX. Dieses Bit ist also aktiv, falls die Quelle XX einen Interrupt sendet.

8.4.3.8.4. Das C40-Interrupt-Maskenregister: C40_INT_MASK

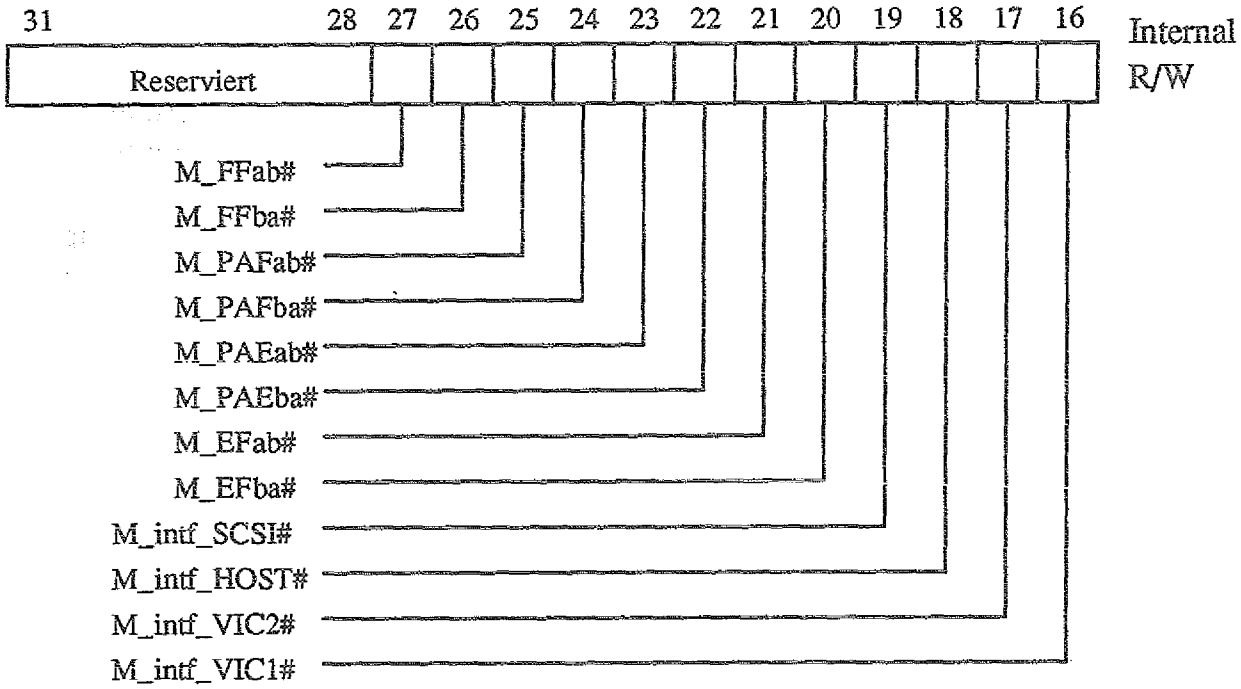


Bild 8-14: Bitpositionen im C40-Interrupt-Maskenregister.

Ein gesetztes Bit im C40-Interrupt-Maskenregister schaltet die jeweilige Interrupt-Quelle ab.
Das Bit M_EFab# bedeutet also: Maskenbit für das Interrupt-Signal EFab# = 0.
EFab# = Empty - Flag (Leerflagge) des BI-FIFOs Richtung a -> b.

8.4.3.8.5. Das C40-Interrupt-Ausgaberegister: C40_INT_OUT

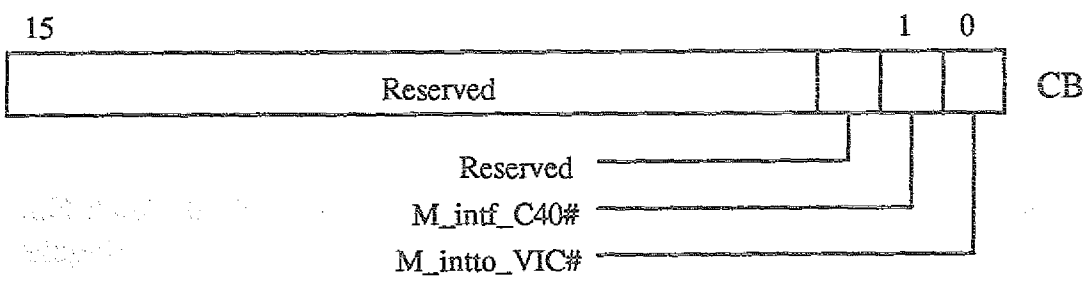


Bild 8-15: Bitpositionen im C40-Interrupt-Ausgaberegister.

Durch das Interrupt-Ausgaberegister wird der C40-Kontaktstift INT_OUT auf genau eine Interrupt-Leitung, entweder zur PCI/PCI-Schnittstelle oder zur VICbus - PCI-Bus Schnittstelle, geschaltet.

Die Bedeutung der Bits im aktiven Zustand ist:

M_intf_C40#: Der C40 aktiviert seine C40_INT_OUT# Leitung und signalisiert einen Interrupt zum Hauptrechner.
M_intto_VIC#: Der C40 aktiviert seine C40_INT_OUT# Leitung und signalisiert einen Interrupt zur VICbus - PCI-Bus Schnittstelle.

Daß beide Bits gleichzeitig aktiv werden, ist sinnlos und sollte unmöglich gemacht werden. Interrupts an die SCSI - Schnittstelle oder an die Speicher - Schnittstelle sind sinnlos, da diese nur als PCI-Slaves agieren können, keine eigene Intelligenz besitzen und daher auch keine Interrupts behandeln können.

8.4.3.8.6. Das C40-Blocklängenregister: C40_BLK_LEN

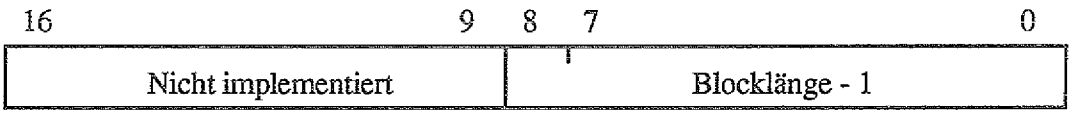


Bild 8-16: Bitpositionen im C40 Blocklängenregister.

In dieses Register ist vor einem Blocktransfer des C40 zum PCI-Bus die Länge des Transfers minus eins einzutragen. Nach einem B_CLEAR#-Signal zählt der Zähler bis zu diesem Wert und beendet durch das B_TC#-Signal (TC = Terminal-Count) den laufenden PCI-Zyklus.

8.4.3.8.7. Das C40-Blockzählerregister: C40_COUNTER

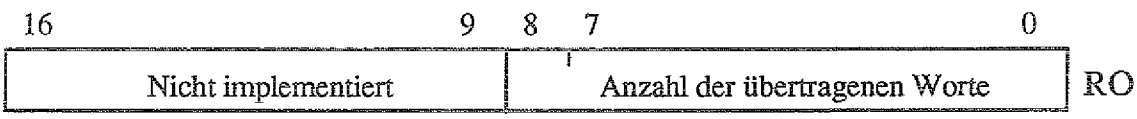


Bild 8-17: Bitpositionen im C40-Blockzählerregister.

Dieses Register ist nur lesbar. Vor einem Blocktransfer ist sein Inhalt gleich Null. Nach einem Blocktransfer steht hier die Anzahl der auf den PCI-Bus übertragenen Worte. Ist dieser Wert von der gesetzten Blocklänge verschieden, kann dadurch ein Fehler festgestellt werden.

8.4.3.8.8. Die FIFO-Konfigurationsregister

Die FIFO-Konfigurationsregister sind in einem Datenblatt der Firma idt (integrated devices technology) dokumentiert /IDT 89/.

9. Literaturverzeichnis

/AMD 93/ N. N., "MACH 3 and 4 Device Families, Data Book", Advanced Macro Devices, Sunyyale, California, Jan. 1993.

/AMEN 89.1/ S. R. Amendolia et al., "Management and Control of the Read Out Processors (TPPs) of the Aleph Time Projection Chamber", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1459 ff., October 1989.

/AMEN 89.2/ S. R. Amendolia et al., "The FASTBUS Read-out System for the Aleph Time Projection Chamber", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1514 ff., October 1989.

/ANDE 92/ Eric C. Anderson et al., "Signal computing", BYTE, Vol. 17, Num. 12, November 1992.

/ANDR 90/ Warren Andrews, "Designers pack intelligence, memory, speed into new DSPs", Computer Design, April 1990.

/ANDR 92.1/ Warren Andrews, "Data acquisition marries DSP on single board", Computer Design, 126, June 1992.

/ANDR 92.2/ Warren Andrews, "DSP boards reach performance highs", Computer Design, pp. 69-79, August 1992.

/ANSI 86/ ANSI/IEEE Std 960-1986, "An American National Standard, IEEE Standard FASTBUS Modular High-Speed Data Acuisition and Control System", The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017, USA, 12 December 1985.

/ANSI 87/ ANSI/IEEE Std 1014-1987. "An American National Standard, IEEE Standard for A Versatile Backplane Bus: VMEbus", The Institute of Electrical and Electronics Engineers, Inc, 345 East 47th Street, New York, NY 10017, USA, 28 March 1988.

/ANTI 90/ F. Antinori et al., "Rejuvenation of a Data Acquisition System for Fixed Target Experiments in a Large Multiuser Spectrometer at CERN", IEEE Transactions on Nuclear Science, Vol.37, No.2, p. 266 ff., April 1990.

/ANTO 89/ I. D' Antone et al., "An Acquisition System Based on a Network of Microvaxes Running the Realtime DEC VAXELN Operating System", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1602 ff., October 1989.

/ANZA 89/ A. Anzalone, "A CAMAC-VME-MACINTOSH DAQ System for Nuclear Experiments", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1671 ff., October 1989.

/ASMU 89/ Riccardo de Asmundis, et al., "On-line DAQ using a Multiprocessor Architecture", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1675 ff., October 1989.

/BALD 91/ M. Baldo-Ceolin et al., "Trigger and Data Acquisition system for the N-Nbar experiment", IEEE Transactions on Nuclear Science, Vol.38, No.2, p. 471 ff., April 1991.

/BAST 91/ C. Bastian et al., "Parallel Processing of Multiparametric Data on a Transputer Array", aus: "IEEE Seventh Conference, REAL TIME '91, on Computer Applications in Nuclear, Particle and Plasma Physics, Conference Record", p. 182 ff., IEEE 1991.

/BAST 92/ T. Bastian et al., "A User Friendly Acquisition System for Cryo-Electron Microscopy", Open Bus Systems '92, p. 179 ff., Zürich, October 1992.

/BEE 88/ P. Bee et al., "The CPLEAR Data acquisition system", aus: "VMEbus in Research, Proceedings of an International Conference, Zürich, Switzerland, 11-13 October 1988", Participants Edition, p. 179 ff., North-Holland 1988.

/BEKE 89/ H. van der Beken, et al., "DAQ at JET - Experience and Progress", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1639 ff., October 1989.

/BELK 89/ A. Belk et al., "DAQ Software Architecture for ALEPH, a Large HEP Experiment", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1534 ff., October 1989.

/BERG 89/ David Berg et al., "A Real Time Integrated Enviornment for Motorola 680xx-based VME and FASTBUS Modules", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1701 ff., October 1989.
-> /PETR 89/

/BETH 90/ Klaus Bethge, Gernot Gruber, "Physik der Atome und Moleküle", Weinheim, VCH Verlagsgesellschaft mbH, 1990.
ISBN 3-527-26933-9

/BLAK 94/ Julian Blake, "Use of LynxOS in the control of CERN's particle accelerators", Online, No.8, January 94, CERN, Geneva, 1994.

/BONA 90/ G. C. Bonazola et al., "The VME Based OBELIX TOF on-line System", IEEE Transactions on Nuclear Science, Vol.37, No.2, p. 315 ff., April 1990.

/BORO 90/ N. Borome, et al., "A VME Multiprocessor Data Acquisition System Combining a UNIX Workstation and Real-Time Microprocessors", aus: "IEEE Transactions on Nuclear Science", Vol. 37, No.4, August 1990, 0018-9499/90/0800-1514$01.00, IEEE 1990.

/BORG 93/ Adriaan J. Borgers, et al., "Data transfer rates using the communication ports of the Texas Instruments TMS320C40 digital signal processor", CERN/EAST not 93-14, September 1993.

/CES 93/ N. N., "CES, Creative Electronic Systems", 1993 Catalogue, Creative Electronic Systems S. A., Petit-Lancy, Geneva, Switzerland, 1993.

/COLO 91/ D. Colombo et al., "The Transputer Based GA.SP Data Acquisition System", aus: "IEEE Seventh Conference, REAL TIME '91, On Computer Applications in Nuclear, Particle and Plasma Physics, Conference Record", p. 190 ff., IEEE 1991.

/DEC 93/ N. N., "DECchip 21050 PCI-to-PCI Bridge Data-Sheet", Order No. EC-N0568-72, Rev. 1.0, Preliminary, December 1993.

/DEC 94/ N. N., "DECdirect, einfach schneller", S. 108, Heft: April 1994.

/DEPP 89/ J. Deppe et al., "ACP/R3000 Processors in Data Acquisition Systems", IEEE Transactions on Nuclear Science, Vol. 36, No.5, p. 1577 ff., October 1989.

/DIJK 88/ J. A. Dijkema, et al., "The VME Event Builder", aus: "VMEbus in Research, Proceedings of an International Conference, Zürich, Switzerland, 11-13 October 1988", Participants Edition, p. 203 ff., North-Holland 1988.

/DIN 87/ Deutsche Norm, DIN 19243 Teil 4, "Grundfunktionen der prozeßrechner-gestützten Automatisierung: Echtzeitfunktionen für Prozeßrechner", Beuth Verlag GmbH Berlin 30, Sep 1987.

/DIST 91/ N. N., "helios", Prospectus of: Distributed Software Limited, Somerset, England, 1991.

/ECK 91/ Chris Eck: "POSIX 1003.4: Standardisation of Real-Time Software", aus: "IEEE Seventh Conference, REAL TIME '91, on Computer Applications in Nuclear, Particle and Plasma Physics, Conference Record", p. 149 ff., IEEE 1991.

/ECKE 87/ Peter Eckelmann, "Transputer der 2. Generation, 1. Teil: Architektur und Merkmale", Elektronik, Heft 18, 1987.

/EDWA 91/ Janet Edwards et al., "Occam and the Transputer- Current Developements", WoTUG-14, 16-18th September 1991, Loughborough, UK, IOS Press, 1991.

/EDWA 92/ John Edwards, "Multi-Processing DSP Systems on the VMEbus", Open Bus Systems '92, p. 211 ff., Zürich, October 1992.

/ENDE 89/ Ch. Ender, "Multiprocessor Data Acquisition System for High Event Rates at the Heidelberg/Darmstadt Crystal Ball", IEEE Transactions on Nuclear Science, Vol. 36, No.5, October 1989.

/ENDE 90/ Ch. Ender, "A New Concept in Data Acquisition Electronics, Control, and Monitoring for the Next Generation of Nuclear Gamma Ray Spectrometer", IEEE Transactions on Nuclear Science, Vol. 37, No.2, April 1990.

/ERVE 92/ W. Erven, J. Holzer, H. Kopp et al., "Intelligent CAMAC Crate Controller with CC-A2 Functionality and VICbus Interface", aus: "IEEE Transactions on Nuclear Science", Vol. 39, Number 4, pp. 853 - 857, August 1992.

/ESON 83/ Esone Committee, "CAMAC, Updated specifications, Volume 1", (c) ECSC-EEC-EAEC, Brusseles Luxembourg, ISBN 92-825-3597-5, 1983.

/ESSE 89/ H. G. Essels, "An Integrated Data Acquisition and Analysis System at GSI", IEEE Transactions on Nuclear Science, Vol. 36, No.5, October 1989.

/ESSE 92/ H. G. Essels, et al. "GOOSY VME System Description - Trigger and Synchronisation", GSI, Gesellschaft für Schwerionenforschung Darmstadt, Germany, May 1992.

/FANO 89/ Penelope Constanta-Fanourakis, et al., "Exabyte Helical Scan Devices at Fermilab", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1696 ff., October 1989.

/FILI 91/ V. Filippini et al., "The Data Acquisition System for the OBELIX Central Detector", aus: "IEEE Transactions on Nuclear Science", Vol. 38, No.2, April 1991, 0018-9499/91/0400-0337$01.00, IEEE 1991.

/FREN 86/ Karen A. Frenkel, "Evaluating Two Massively Parallel Machines", Com. of the ACM, Vol. 29, No. 8, August 1986.

/FULG 89/ W. Fulgione et al., "The LVD Experiment Data Acquisition System", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1635 ff., October 1989.

/GEE 91/ David M. Gee et al., "Devlopment methods and occam", p. 111 ff., WoTUG-14, 16-18th September 1991, Loughborough, UK, IOS Press, 1991.

/GEES 89/ D. F. Geesaman et al., "Data Acquisition for FNAL E665", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1528 ff., October 1989.

/GM 88/ N. N., "Manufacturing Automation Protocol - Version 3.0", General Motors Corporation, 1988.

/HAGE 93/ J. Hagedorn, J. Holzer, H. Kopp, K. Zwoll, "Hochgeschwindigkeits-PCI-VICbus Interface, V 1.0", Forschungszentrum Jülich, Zentrallabor für Elektronik, Abt. Experimentsteuerung und Kommunikation, IB-KFA-ZEL 501393, 4. November 1993.

/HARR 91/ F. Harris: "Design and Performance of The DELPHI Data Acquisition System", aus: "IEEE Seventh Conference, REAL TIME '91, on Computer Applications in Nuclear, Particle and Plasma Physics, Conference Record", p. 101 ff., IEEE 1991.

/HATZ 88/ Despina Hatzifotiadou, "OPAL Data Acquisition" , aus: "VMEbus in Research, Proceedings of an International Conference, Zürich, Switzerland, 11-13 October 1988", Participants Edition, p. 185 ff., North-Holland 1988.

/HAYN 88/ W. J. Haynes, "H1 VME Data Acquisition", aus: "VMEbus in Research, Proceedings of an International Conference, Zürich, Switzerland, 11-13 October 1988", Participants Edition, p. 191 ff., North-Holland 1988.

/HOAR 78/ C. A. R. Hoare, "Communicating Sequential Processes", in: Communications of the ACM, Vol. 21, No. 8, Aug. 1978.

/HOAR 85/ Charles Antony Richard Hoare, "Communicating Sequential Processes", Prentice-hall International UK, LTD, ISBN-0-13-153271-5, 1985.

/HOLZ 92.1/ Jürgen Holzer et al., "TURBOchannel VICbus-Interface", Open Bus Systems '92, Zürich, Proceedings, p. 409-412, October 1992.

/HOLZ 92.2/ Jürgen Holzer et al., "VICbus-VSB-Interface", Internal Report IB-KFA-ZEL 500592, Forschungszentrum Jülich, 1992.

/HOLZ 93/ Jürgen Holzer, Klaus Zwoll, "TURBOchannel Interface", interner Bericht, Forschungszentrum Jülich GmbH, März 1993.

/HULS 91/ John Hulskamp, editor, et al., "The Transputer in Australasia 2", ATOUG-4, Proceedings of the 4-th Austrailian Transputer and OCCAM User Group Conference, 23-24 Sept. 1991, Canberra, Australia.

Diese Veröffentlichung enthält folgende Artikel:

/Bos/ Terry Bossomaier, Adrian Loeff, "Portable C* for MIMD Machines", Computer Science Lab., Australian National Uni.

/Gra/ J. P. Gray, I. E. Jelly, "Using OCCAM and Transputers to Simulate a Multiprocessor Architecture with Broadcast facilities". jon@cs.uow.edu.au, innes@cms.scp.ac.uk

/Kaw/Toshiyuki Kawai, et al., "Ray Tracing for Parallel Image Generation System MAGG", Japan.

/Moh/G. M. Mohay, A. Seagrott, "DECOP: A Development Enviornment for Configuring occam Programs", Queensland.

/Phi/ L. Phillips, et al., "Target Transputer System for Undergraduate Coursework", Aus. Defence Force Academy, Campbell.

/Shi/ Xuehua Shi, et al., "Transputer Applications in Real Time Tomography", Aus.

/Sto/ F. H. Stootman, S. Hansen, "The Transputer as an Engineering Device".

/Tab/ Nozar Tabrizi, Fazel Naghdy, "Low Cost I/O TRAMs for Real-Time Applications", Uni. of Wollongong, Aus.

/Tia/ Y. Tian, et al., "A Tool for Fault Prevention in Transputers", Uni of Waikato, New Zealand.

/Zha/ Kang Zhang, "Towards Occam Program Visualisation", Brighton, UK.

/HUTH 93/ Norbert Huth, "VL-Bus oder PCI oder was?", Systeme, S. 46-53, Heft 5/93.

/ICHI 89/ T. Ichihara et al., "Data Acquisition System at the RIKEN Accelerator facility", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1628 ff., October 1989.

/IDT 89/ N. N., "High-Speed CMOS Data Book", Integrated Devices Technology, Inc., Santa Clara, California, USA, 1989.

/INMO 89/ N. N., "Technical note 5: Programm design for concurrent systems", The transputer applications notebook - Systems and performance, INMOS Limited, 1989.

/INMO 91/ N. N., "The T9000 Transputer Products Overview Manual", First Edition, Order Code: DBTRANSPST/1, INMOS Limited, 1991.

/INMO 92.1/ N. N., "General Purpose MIMD Machines, High Performance Heterogeneous Interprocessor Communication", ESPRIT Project 5404, DS-Links, Technical Information, SGS-THOMSON Microelectronics, November 1992.

/INMO 92.2/ N. N., "Transputer Databook", Third Edition INMOS Limited, Order Code: DBTRANST/3, 1992.

/INMO 92.3/ N. N., "T9000 Transputer Hardware Reference Manual", Instruction and data cache, Preliminary Information, INMOS Limited, Januar 1992.

/INTE 93/ PCI Special Interest Group (SIG), "PCI Local Bus Specification", Product Version, Revision 2.0, PCI SIG, Hillsboro, OR, USA, April 30, 1993.

/INTE 94/ N. N., "iFX8160 FLEXlogic FPGA with SRAM Option", INTEL CORPORATION, Order Number: 290461-004, March 1994.

/ISO 90/ N. N., "ISO9506 - Manufacturing Message Specification", International Organization of Standardization, 1990.

/ISO 93/ N. N., "VICbus, Inter-Crate Bus for the IEC821 VMEbus", ISO/IEC 26.11458 Draft Specification, Revision 2.0, International Organisation for Standards International Electrotechnical Commission, Joint Technical Committee 1, Sub-Committee 26, Working Group 8, March 24, 1993.

/JOHN 89/ Harald Johnstad, "Physics Analysis Workstation", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1568 ff., October 1989.

/KALS 88/ M. Kallstrom, S. S. Thakkar, "Programming Three Parallel Computers", IEEE Software, Vol. 5, No. 1, pp. 11-22, Jan. 1988.

/LANG 88/ Herbert Lang, "A Racetrack for Cool Particles, COSY Jülich", Kernforschungsanlage Jülich GmbH, COSY-Projektleitung, 1988.

/LAWS 92/ Harold W. Lawson, "Parallel Processing in Industrial Real-Time Applications", Prentice Hall, Englewood Cliffs, New Jersey, USA, 1992. ISBN 0-13-654518-1

/LSI 93.1/ N. N., "MDC40S TIM-40 Module, User Manual", Version 1.02 Loughborough Sound Images Ltd., England, May 1993.

/LSI 93.2/ N. N., "QPC/C40B, User Guide", Version 1.00 Loughborough Sound Images Ltd., England, June 1993.

/LSI 93.3/ N. N., "QPC/C40B, Technical Reference Manual", Version 1.00 Loughborough Sound Images Ltd., England, June 1993.

/MAIE 93/ Max Maier, "PCI and MPI Bus", MM-CERN/SCI Workshop PCI/MPI Bus, 15 September, 1993.

/MANO 85/ Tamio Mano et al., "OCCAM to CMOS, Experimental Logic Design Support System", Computer Hardware Description Languages and their Applications, C. J. Koomen and T. Moto-oka (eds.), Elsevier Science Publishers B. V. (North-Holland), IFIP, 1985.

/MERK 91/ Martin Merkel: "The Crystal Barrel Data Acquisition System", aus: "IEEE Seventh Conference, REAL TIME '91, on Computer Applications in Nuclear, Particle and Plasma Physics, Conference Record", p. 123 ff., IEEE 1991.

/MILD 85/ Johannes P. Milde, "Überlegungen zur Organisation verteilter Mehrrechnersysteme", Dissertation an der RWTH Aachen, Fakultät Elektrotechnik, 1985

/MOEL 94/ Julie Moeller-Buchner, "Top-Quark-Entdeckung, Meilenstein der Physik", Zeitungsartikel in den Dürener Nachrichten vom 26. April 1994.

/MOLE 89/ A. Vander Molen et al., "Status of the NSCL 4PI Data Acquisition System", IEEE Transactions on Nuclear Science, Vol. 36, No.5, p. 1558 ff., October 1989.

/MUSI 88/ Gerhard Musiol, et al., "Kern- und Elementarteilchenphysik", Weinheim, VCH Verlagsgesellschaft mbH, 1988. ISBN 3-527-26886-3

/NASS 93/ Richard Nass, "I/O Tasks Find Their Way Onto The PCI Bus", Electronic Design, October 14, 1993.

/NCR 93/ N. N., "NCR 53C810, PCI-SCSI I/O Processor, Data Manual", NCR Corporation, Ohio, USA, 1993.

/NINA 92/ A. Ninane, et al., "The Design of a VMEbus General-Purpose Data Acquisition System", Conference Paper, Open Bus Systems '92, Zürich, October 1992.

/OELE 92/ W. Oelert et al., "Proposal for an Experiment at COSY Jülich, Threshold Meson Production at the Internal COSY-Beam in the Range of Scalar Mesons Involving Strangeness", COSY Proposal/Letter of Intent, Experiment No.: COSY 11, 1992.

/PARK 93/ C. Parkman, "VICbus - Tutorial", Online, No.6, April 93, No.7 Juli 93, No.8 Januar 94, CERN, Geneva, 1993.

/PARS 94/ N. N., "Parallel Power for Industry", Prospekt der Firma Parsytec, Aachen, 1994.

/PAST 92/ Antonio Pastore, "The HIPPI to TURBOchannel Connection", Open Bus Systems '92, p. 71 ff., Zürich, October 1992.

/PETE 92/ J. Petersen, "VMEbus Based Data Acquisition with Real-Time UNIX on CISC and RISC Processors - Some Performance Measurements", Open Bus Systems '92, p. 99 ff., Zürich, October 1992.

/PETR 89/ Donald L. Petravick et al., "The PAN-DA Data Acquisition System", IEEE Transactions on Nuclear Science, Vol. 36, No.5, p. 1540 ff., October 1989.
-> /BERG 89/ /PORD 89/

/PLES 86/ Klaus W. Pleßmann und Charalambos Tassakos, "Die Programmiersprache OCCAM", Angewandte Informatik, Heft 9 1986, Vieweg Verlag.

/PONT 91/ P. Pontin, H. Verweij: "Instrumentation Busses for High Energy Physics, Past, Present and Future", IEEE Transactions on Nuclear Science, Vol. 38, No.2, p. 322 ff., April 1991.

/PORD 89/ Ruth Pordes et al., "Software for FASTBUS and Motorola 68K Based Readout Controllers for Data Acquisition", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1544 ff., October 1989.

/QUIN 90/ S. Quinton et al., "A Review of the Data Acquisition Electronics for the ISIS Spallation Neutron Source", IEEE Transactions on Nuclear Science, Vol.37, No.6, p. 2156 ff., December 1990.

/ROST 91/ M. Rost und W. Weihs, "Transputer Based DAQ System for 100 Mbyte/sec Transfers", aus: "IEEE Seventh Conference, REAL TIME '91, on Computer Applications in Nuclear, Particle and Plasma Physics, Conference Record", p. 432 ff., IEEE 1991.

/RUED 89/ Wolfgang von Rüden, "The ALEPH Data Acquisition System", IEEE Transactions on Nuclear Science, Vol.36, No.5, p. 1444 ff., October 1989.

/SCHM 88/ Hans von der Schmitt et al., "From FADC Data to Track Parameters in OPAL's VME System", aus: "VMEbus in Research, Proceedings of an International Conference, Zürich, Switzerland, 11-13 October 1988", Participants Edition, p. 211 ff., North-Holland 1988.

/SCHN 93.1/ Georg Schnurrer, "Moderne PC-Bussysteme", c't, S. 110-119, Heft 10, 1993.

/SCHN 93.2/ Georg Schnurrer, "PCI-Almanach", c't, S. 32-34, Heft 11, 1993.

/SCHN 94.1/ Georg Schnurrer, "Kraftakt, Im Test: PCI-Boards mit 486- und Pentium CPU, PCI-Hostadapter und eine PCI-Netzwerkkarte", c't, S. 68-86, Heft 2, 1994.

/SCHU 88/ Alois Schütte, "Programmieren in OCCAM", Addison-Wesley Verlag, ISBN 3 925118 56 X, 1988.

/SPEC 91/ N. N., "SPOX, The DSP Operating System, A Technical Overview", Spectron Microsystems Incorporated, 1991.

/STRU 92/ Struck, "STR330/VIC', Technical Manual, Firma Struck, Hamburg, Germany, 1992.

/TI 91.1/ N. N., "Texas Instruments, TMS320C4x User's Guide", 2564090-9721 revision A, May 1991.

/TI 91.2/ N. N., "TMS320 Floating-Point DSP Optimizing C Compiler", 2576391-9721 revision A, October 1991.

/TI 91.3/ N. N., "TMS320 Floating-Point DSP Assembly Language Tools", 2576328-9721 revision A, September 1991.

/TI 92.1/ N. N., "Texas Instruments, TMS320, DSP New Products Seminar", TI Schulungszentrum für Mikroelektronik, München, 1992.

/TI 92.2/ N. N., "TMS320 Familiy, Development Support Reference Guide", Texas Instruments, 1992.

/TI 92.3/ N. N., "TMS320C4x C Source Debugger", Texas Instruments, April 1992.

/TI 92.4/ N. N., "TMS320C4x Parallel Runtime Support Library User's Guide", 2617613-9761, Texas Instruments, Sept. 1992.

/TI 93.1/ TIM-40 Coordinator, "TIM-40, TMS320C4x Module Specification", Version 1.01, Texas Instruments, January 1993.

/TI 93.2/ N. N., "Software Development Tools Technical note for: TMS320", Texas Instruments, 3rd Quarter 1993.

/VERM 93/ J. C. Vermeulen, NIKHEF-H, "Data transport with the Texas Instruments TMS320C40 DSP", EAST note 93-21, 18 October 1993. EAST93-21

/VIEW 92.1/ N. N., "Powerview 5.1 Release Notes", Viewlogic Systems Inc., Marlboro, MA, USA, 1992.

/VIEW 92.2/ N. N., "Powerview Installation", VIEWlogic Systems Inc., Marlboro, MA, USA, 1992.

/VIEW 92.3/ N. N., "Digital Design Tutorial", VIEWlogic Systems Inc., Marlboro, MA, USA, 1992.

/VIEW 92.4/ N. N., "Occurrence AttribuTe (OAT) Tutorial", VIEWlogic Systems Inc., Marlboro, MA, USA, 1992.

/VIEW 92.5/ N. N., "VHDL Tutorial", VIEWlogic Systems Inc., Marlboro, MA, USA, 1992.

/VIEW 92.6/ N. N., "ViewDoc User's Guide", VIEWlogic Systems Inc., Marlboro, MA, USA, 1992.

/VIEW 92.7/ N. N., "ViewSim Reference Tutorial", Viewlogic Systems Inc., Marlboro, MA, USA, 1992.

/VIEW 92.8/ N. N., "PCB interface users guide", Viewlogic Systems Inc., Marlboro, MA, USA, 1992.

/VILL 91/ Umberto Villano, "Repeatable Execution of Occam Programs", p. 133 ff., WoTUG-14, 16-18th September 1991, Loughborough, UK, IOS Press, 1991.

/WIES 92/ Johann Wiesböck, Hr., "DSPs in der Praxis: Architekturen Werkzeuge, Software, Applikationen", 16./17. Juni 1992, München, Markt & Technik Verlag AG, 1992.

Diese Veröffentlichung enthält folgende Artikel:

/Hel/ Manfred Helze, "Digitaler Signalprozessor unterstützt Parallelverarbeitung", Hema Systemknowhow, S. 179-182.

/Kel/ Cornelius Kellerhoff, "Multiprocessortopologien mit dem TMS320C30", Electronic Tools, S. 183-185, 1992.

/Aug/ Gisbert Auge, Norbert Witt, "Synchrone, parallele Signalverarbeitung", Systemforschung.

/Mey/ Bern Meyer, "Software für Grafikprozessoren", Texas Instruments, S. 191-194, 1992.

/Dum/ Ulrich Dumschat, "Optische Qualitätssicherung mit Parallel-Prozessoren (TMS320C40, Transputer)", S. 221-224, hema-Elektronik, Aalen, 1992.

/Jou/ Burkard Jour, "Harte und weiche Ware für den TMS320C40, oder: von der PC-Karte zum DSP-Betriebssystem", Electronic Tools, Ratingen, S. 265-278, 1992.

/Kel/ Cornelius Kellerhoff, "Echtzeitsignalverarbeitung auf VMEbus-Basis", Eletronic Tools, Ratingen, S. 279-289, 1992.

/Chr/ Manfred Christ, "Hochsprachen-Debugger für den DSP TMS320C30", Texas Instruments, 1992.

/WILS 92/ Dave Wilson, "Is RISC or DSP best for your application?", Computer Design, pp. 65-72, April 1992.

/WUES 92/ Peter Wüstner et al., "Cosy Data Acquisition, Kommunikation: Struktur, Control- Datenfluß", Internes Material des Zentrallabors für Elektronik dès Forschungszentrums Jülich, 1992.

/ZOLL 91/ J. Zoll, "Zebra Reference Manual, book FZ, Zebra version 3.66", CERN Geneva, 1991.

/ZWOL 91/ Klaus Zwoll et al., "COSY-Experiment Datenerfassung, Systemübersicht V1.1", Berichte zum Datenerfassungssystem für physikalische Experimente an COSY, Forschungszentrum Jülich, Interner Bericht KFA-ZEL-IB-500891, 1991.

/ZWOL 92/ Klaus Zwoll et al., "Architektur und Übersicht des Datenerfassungssystems für COSY-Experimente", Draft 1.1 des Zentrallabors für Elektronik, Abt. Experimentdatenverarbeitung und Kommunikation, Forschungszentrum Jülich, 1992.

/ZWOL 93/ Klaus Zwoll et al., "Objektorientierte Modellierung des On-Line-Datenerfassungssystems für COSY-Experimente, V 2.0", Forschungszentrum Jülich, Zentrallabor für Elektronik, Abt. Experimentsteuerung und Kommunikation, IB-KFA-ZEL 501293, 18. November 1993.

/ZWOL 94/ K. Zwoll, M. Drochner, W. Erven, J. Holzer, H. Kopp et al., "Flexible Data Acquisition system for experiments at COSY", aus: "IEEE Transactions on Nuclear Science", Vol. 41, Number 1, pp. 37 - 44, Febuary 1994.

/3L 92/ N. N., "Parallel C User Guide", 3L Ltd., Edinburgh, Scotland, Support@ThreeL.Co.UK, 1992.

/3L 93.1/ N. N., "3L C40 Technical Note 0, Index to C40 Technical Notes", 3L Ltd., 03700 October 8, 1993.

/3L 93.2/ N. N., "3L C40 Technical Note 1, C40 Hardware Guidelines", 3L Ltd., 03701 October 8, 1993.

/3L 93.3/ N. N., "3L C40 Technical Note 9, Using Unsupported C40 Hardware", 3L Ltd., 03709 September 13, 1993.

/3L 93.4/ N. N., "3L C40 Technical Note 26, Configuration Guide for the LSI QPC/C40", 3L Ltd., 03726 August 27, 1993.

/3L 93.5/ N. N., "3L C40 Technical Note 27, Handling Interrupts in Parallel C", 3L Ltd., 03727 October 19, 1993.

/3L 93.6/ N. N., "3L C40 Technical Note 28, Writing a Device Driver: an Example", 3L Ltd., 03728 October 19, 1993.

/3L 93.7/ N. N., "3L C40 Technical Note 29, A DSP Example: Real-Time Filtering", 3L Ltd., 03729 October 19, 1993.

/3L 93.8/ N. N., "3L C40 Technical Note 31, Parallel Real-Time FIR Filtering", 3L Ltd., 03731 October 19, 1993.

/3L 93.9/ N. N., "C40 Debugger Support Kit V1.0, Release Note", 3L Ltd., June 4, 1993.

/3L 93.10/ N. N., "Parallel C Version 1.0.1, Texas Instruments TMS320C40, Release Note", 3L Ltd., October 18, 1993.

/3L 93.11/ N. N., "Parallel C, C40 Library Extensions, User Guide", 3L Ltd., October 13, 1993.

FORSCHUNGSZENTRUM JÜLICH GmbH

KFA

Jül-3013
Januar 1995
ISSN 0944-2952