Ein Mikroprozessor als Trigger für geladene Teilchen bei H1

Diplomarbeit an der Universität Hamburg

Fachbereich Physik

Wilhard von Wendorff

Betreuender Professor: H. Spitzer[1]
Betreuer für Rechnerfragen: H.-J. Stuckenberg[2]
Betreuer für physikalische Fragen: H.-J. Behrend[3]

25. April 1988

[1]II. Phys. Institut der Universität Hamburg
[2]F56 DESY–Hamburg
[3]F36 DESY–Hamburg

Inhaltsverzeichnis

1 Einleitung 4
1.1. Der H1–Detektor 5
1.2. Das Triggerproblem 7
1.3. Inhalt der Arbeit 8

2 Apparative Gegebenheiten 10
2.1. Die zentrale Driftkammer 10
2.2. Die Datenauslese der zentralen Driftkammer 12

3 Trigger bei H1 15
3.1. Die Triggerstufen 15
3.2. Die geplanten Triggerkomponenten bei H1 19
3.3. Notwendigkeit eines Driftkammertriggers 19
3.4. Die Untergrundereignisse 27

4 Verschiedene Triggeralgorithmen ("tracking"- Algorithmen) 31
4.1. Spursuche unter Verwendung der Kreisgleichung 31
4.2. Spursuche in Baumstrukturen 35
4.2.1. Das Aufbauen der Verbindungspaare 36
4.2.2. Das Sortieren der Liste der Verbindungspaare 37
4.2.3. Das Aufbauen der Ketten 38
4.2.4. Eigenschaften der Spursuche in Baumstrukturen 39
4.3. Spursuche im reziproken Raum 39
4.4. Spursuche mit einem "Masken"- Prozessor 42
4.5. Begrenzter *Football*- Algorithmus 46
4.6. Zusammenfassung 48

5 Verschiedene Prozessortypen 49
5.1. Der Digitale Signalprozessor DSP56001 von Motorola 49
5.2. Der Prozeßrechner MC68020 von Motorola 49
5.3. Transputer von Inmos (IMS T414) 50

6 Verwendetes Programm (für den DSP56001) 52
6.1. Die assoziative Speicher 52
6.1.1. Inhaltsadressierte assoziative Speicher (CAM) 52
6.1.2. Die Simulation der CAMS 54
6.2. Die Tabellen ("lookuptables") 55
6.3. Das Assemblerprogramm 56

7 Die Simulations des Triggersystemes **61**
7.1. Die verwendeten Monte–Carlo Daten 61
7.2. Physikalische Eigenschaften der Monte–Carlo Daten 64
7.3. Die Parameter der Simulation 68
7.4. Simulationsablauf 72
7.5. Die Ergebnisse der Simulation 74

8 Die Kosten eines solchen Triggers **82**
8.1. Die Kosten der Prozessorkarte 82
8.2. Die Kosten der CAM–Karten 82
8.3. Die Kosten der Schieberegister–Karten 83
8.4. Die Kosten der Winkelberechnungseinheit 84
8.5. Die Gesamtkosten 84

9 Möglichkeiten zur Erweiterung eines solchen Triggers **87**
9.1. Parallele Ereignisverarbeitung 87
9.2. Parallele Winkelsegment- Verarbeitung 88
9.3. Parallele Spurverarbeitung 89
9.4. Parallele Verarbeitung von Spursegmenten 89
9.5. Zusammenfassung 90

10 Schlußbetrachtung **91**

Anhang **92**

I Der Digitale Signalprozessor DSP56001 **92**
I.1. Beschreibung des DSP56001 92
I.2. Zukünftige Verbesserungen des DSP56001 97

II Der Universalprozessor MC68020 **98**
II.1. Beschreibung des MC68020 98
II.2. Zukünftige Verbesserungen des MC68020 100

III Des Transputer IMS T414 **101**
III.1. Beschreibung des IMS T414 101
III.2. Weiterentwicklungen der Transputer 106

IV Die Simulation **107**
IV.1. Ein Beispiel 107

V Programme für den DSP56001 **109**
V.1. Flußdiagramm von Assemblerprogramm 109
V.2. Das Assemblerprogramm 133

VI FORTRAN Programme **158**
VI.1. Simulation des Maskenprozessors 158
VI.2. Programm für Kreisgleichungsalgorithmus 165

Abbildungsverzeichnis **193**

Index **195**

Dank **198**

Erklärung **198**

Parameter :	Protonenring	Elektronenring	Einheit
Nominelle Strahlenergie	820	30	GeV
Schwerpunktsenergie	315	315	GeV
Einschußenergie	40	14	GeV
Anzahl der Wechselwirkungspunkte	4	4	
Luminosität	$1.5 \cdot 10^{31}$	$1.5 \cdot 10^{31}$	cm^2s^{-1}
Umfang	6335	6335	m
Umlauffrequenz	47.31	47.31	kHz
max. Anzahl der Teilchenpakete	220	220	
Teilchenstrom	163	58	mA
Teilchenanzahl pro Paket	$1 \cdot 10^{11}$	$3.5 \cdot 10^{10}$	
horizontale Emittanz (σ_x)	$0.86 \cdot 10^{-8}$	$3.5 \cdot 10^{-8}$	π rad
vertikale Emittanz (σ_z)	$0.43 \cdot 10^{-8}$	$0.69 \cdot 10^{-8}$	π rad
Länge des Teilchenpakets	110	7.8	mm
Zeitabstand zwischen Teilchenpakete	96	96	nsec
Energieverlust pro Umlauf	$1.4 \cdot 10^{-10}$	71.9	MeV
Hochfrequenz	208.194	499.6655	MHz
Füllzeit	20	15	min
Stärke der Magneten	4.649	0.1426	T
Vakuum	$< 10^{-8}$	$< 10^{-8}$	mbar

Tabelle 1: Parameter von HERA

1 Einleitung

Bei DESY–Hamburg wird bis ca. 1989 ein neuer Speicherring (HERA) in Betrieb genommen. Dieser besteht aus einem 30GeV Elektronenring und einem 820GeV Protonenring (siehe Tabelle 1). HERA soll die Untersuchung der Elektron– Proton–Wechselwirkung ermöglichen, z.B. die Untersuchung von Elektron–Quark–Streuung, Photon–Gluon–Streuung und W–Gluon–Streuung (Siehe Abbildung 1). Diese Untersuchungen sollen Antworten liefern zu folgenden vier Fragen:

- Welche innere Struktur besitzen Protonen, Elektronen und Quarks?
- Welche Eigenschaft und Struktur weist die Elektro–Schwache–Wechselwirkung auf?
- Welche Eigenschaft und Struktur weist die Starke–Wechselwirkung auf?
- Gibt es neue Teilchen wie Lepto–Quarks und Top–Quarks?

Die Topologie der Ereignisse mit neutralen (englisch: neutral current, NC) und geladenen Strömen (englisch: charged current, CC) sind in der Abbildung 1 dargestellt. Das Lepton und der Jet geladener Teilchen (mehrere Jets falls auch Gluonen

frei werden) emittieren auf entgegengesetzten Seiten der Strahlachse. So bleibt die Summe der Transversalimpulse Null. Die Zerfallsprodukte des Protons bilden einen schmalen Konus in Protonenrichtung mit einem Öffnungswinkel von einigen mRad.

1.1. Der H1–Detektor

Eines der Experimente an dem Speicherring HERA ist H1. Dieser Detektor besteht aus verschiedenen Komponenten (siehe auch Abbildung 2):

Jet- Kammer: eine zentrale Jet- Driftkammer {1} (englisch: central jet chamber, CJC) zur Rekonstruktion der Spuren geladener Teilchen, unterbrochen durch Z-Kammern und Proportionalkammern (MWPC). Im Vorwärtsbereich befinden sich eine Serie von radialen und planaren Driftkammern die durch drei Lagen Proportionalkammern und Radiatoren für Übergangsstrahlung (englisch: transition radiators) {2} unterbrochen werden.

EM Kalorimeter: ein elektromagnetisches Kalorimeter, aufgebaut aus Flüssigargon und Bleiabsorberplatten im Vorwärts- und Zentral- Bereich {3} und ein Bleiabsorber–Scintillator– Kalorimeter im Rückwärtsbereich {5}.

Hadron Kalorimeter: ein aus Flüssigargon und Eisenabsorberplatten aufgebautes Kalorimeter für hadronische Teilchen {4}.

Feldspule: eine supraleitende Spule und Kryostat{6} außerhalb des Hadron- Kalorimeters. Diese Spule erzeugt ein möglichst homogenes magnetisches Feld in der zentralen Driftkammer. Das Eisenjoch dieser Spule dient zusätzlich zur Abschirmung der Myonkammern vor Hadronen.

Eisenplatten: ein mehrwandiger Mantel aus Eisenplatten {7}. Zwischen diesen Platten befinden sich Kunstoff- "streamer"- Rohre, welche Restenergien von Hadronen, die das Kalorimeter verlassen, bestimmen (englisch: tailcatcher). Zur Rekonstruktion von Myonspuren und als Myonfilter ist diese Detektorkomponente ebenfalls geeignet.

Myon- Detektor: in Vorwärts- und zentraler Richtung befinden sich drei Lagen Myonkammern {9} und zusätzlich in Vorwärtsrichtung ein Myon- Spektrometer aus einem Magneten {8} und vier Lagen Driftkammern {9}.

Klein- Winkel- Kalorimeter: ein eingeschobenes Kalorimeter {10}, um hadronische Energien bei kleinen Vorwärtswinkeln (Winkel zur Protonenstrahlrichtung) zu messen. Es können Winkel bis 0,7° erfaßt werden.

Korrekturmagnet: ein Korrekturmagnet zur Kompensation von radialen Anteilen der großen supraleitenden Magnetspule.

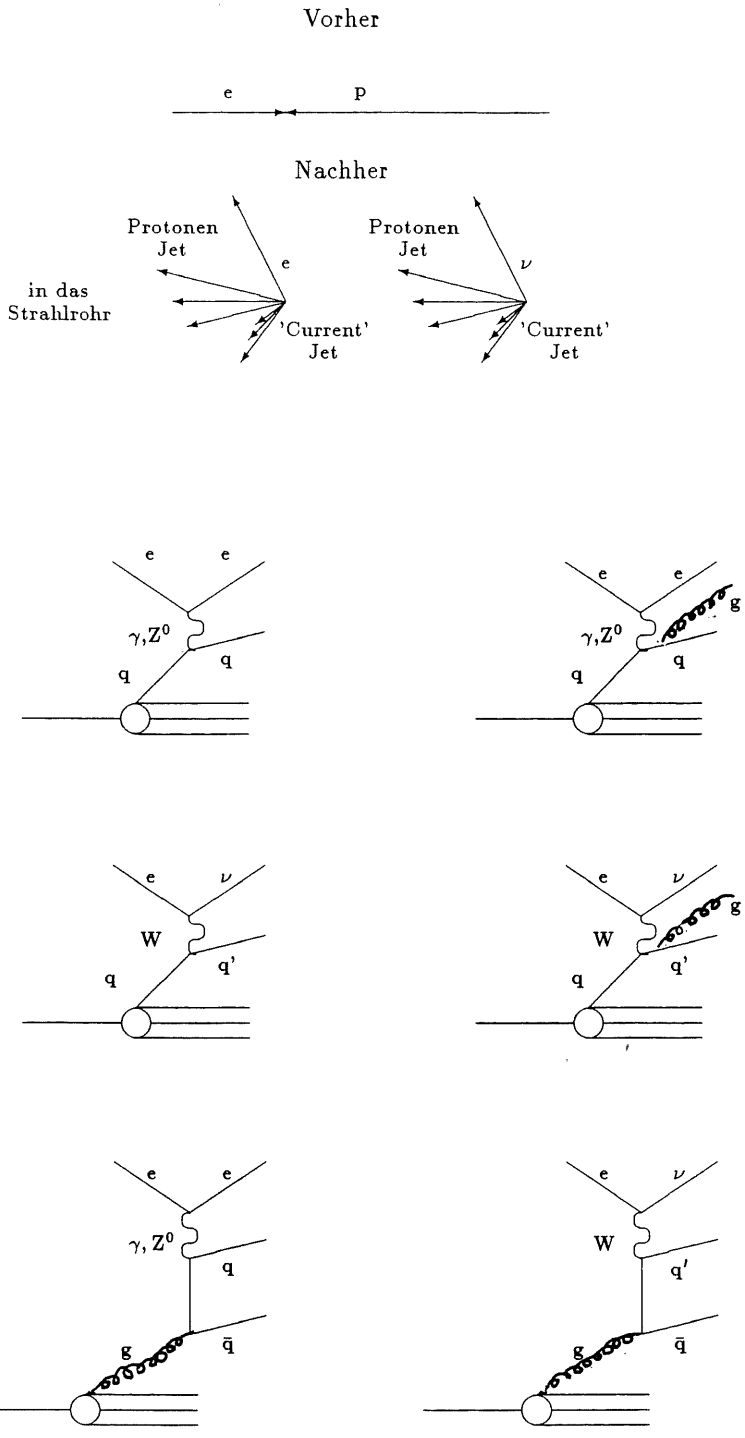


Abbildung 1: Topologie von *ep* Ereignisse

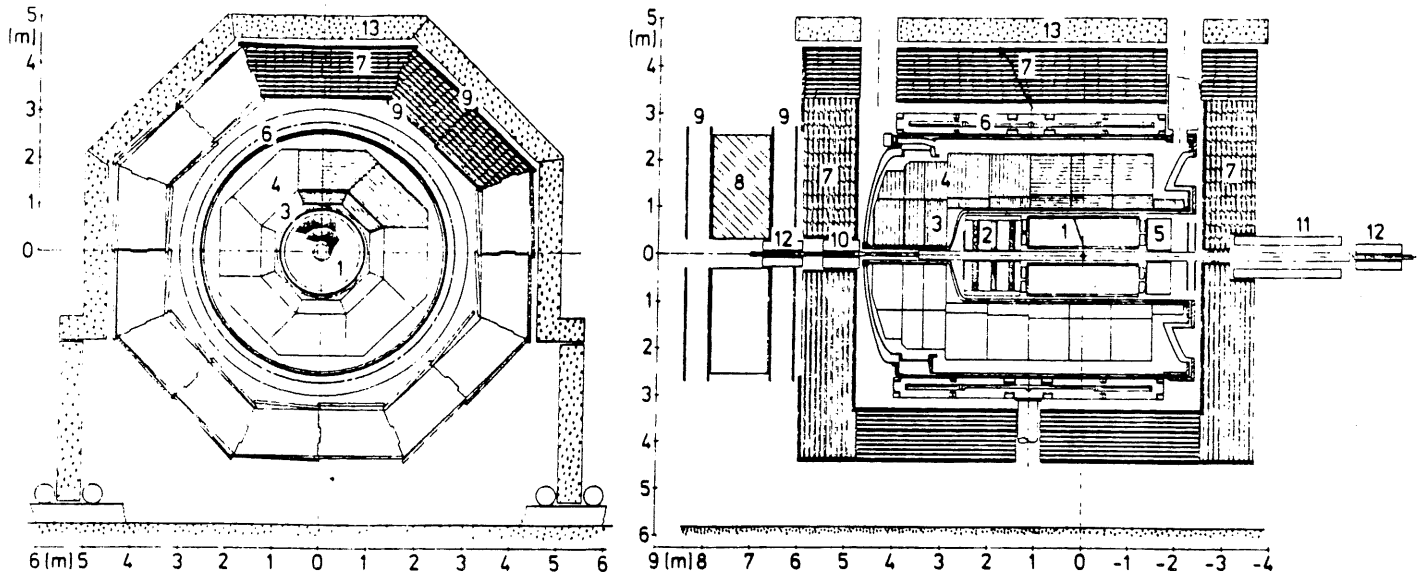


Bild 1: Zwei Querschnitte der H1-Apparatur. 1: zentrale Spurenkammern, 2: Vorwärts-Spurenkammern und Detektoren für Übergangsstrahlung, 3: elektromagn. Flüssig-Argon-Kalorimeter, 4: hadronisches Flüssig-Argon-Kalorimeter, 5: Rückwärts-Kalorimeter (elektromagn.), 6: supraleitende Spule (Solenoid), 7: Eisen mit Nachweiskammern, 8: toroidales Eisenjoch, 9: Myonenkammern, 10: Pfropfen-Kalorimeter, 11: Kompensationsspule (Solenoid), 12: HERA-Quadrupol, 13: Beton.

Abbildung 2: Aufbau des H1-Detektors

1.2. Das Triggerproblem

Die modernen Detektoren wie H1, welche aus vielen verschiedenen Detektorkomponenten mit einigen zehntausend Kanälen bestehen, erlauben es, hohe Ereignisraten aufzulösen. Da jedes Ereignis eine Datenmenge von einigen Millionen Bits erzeugt und die Übertragung und Speicherung eines Ereignisses auf einem Großrechner einige $\frac{1}{10}$ Sekunden benötigt, besteht die Notwendigkeit die Ereignisrate zu senken. Der bei DESY installierte Großrechner (IBM3084Q) kann maximal eine Ereignisrate von fünf Hertz speichern. Bei HERA kreuzen sich die Teilchenpakete (englisch: bunch crossing) jedoch alle 96nsec. Wie in Tabelle 3 zu sehen ist, tritt im zeitlichen Mittel bei jeder 30. Kreuzung (300kHz$\times$96nsec) eine Wechselwirkung eines Teilchenpaketes mit anderen Teilchen (Restgas, Proton, usw.) auf, welche dann als physikalisch relevantes und irrelevantes Ereignis erkannt werden muß. Die Rate der relevanten Ereignisse ist auf Grund des kleinen Wechselwirkungsquerschnittes der schwachen und elektromagnetischen Wechselwirkung um den Faktor 10^3 niedriger als die physikalisch irrelevanter Ereignisse, welche im allgemeinen auf der starken Wechselwirkungen des Strahls mit Restgasatomen oder Strahlrohrmaterie beruhen. (Die verschiedenen zu erwartenden Ereignisraten sind ausführlich in Tabelle 2 und 3 aufgeführt). Daher müssen die irrelevanten Ereignisse so früh wie möglich aussortiert werden. Die unterschiedlichen Eigenschaften der Spuren geladener Teilchen und der deponierten Energie im Kalorimeter bei physikalisch relevanten und irre-

levanten Ereignissen müssen für die Konstruktion von Triggern verwendet werden (z.B. kommen die Spuren relevanter Ereignisse vom Wechselwirkungspunkt). Ein Trigger direkt an der Datenauslese eines Detektors sollte in der Lage sein, auf Grund dieser Eigenschaften schnell zwischen relevanten und irrelevanten Ereignissen unterscheiden zu können.

Seit bei modernen Detektoren zur Hauptsache Driftkammern zur Spur-Rekonstruktion geladener Teilchen verwendet werden, stehen den Triggern eine große Menge Signale für die Triggerentscheidung zur Verfügung. So können Spuren von Teilchen schnell durch den Vergleich der Signale der Driftkammern mit einer Bibliothek von einigen tausenden Spuren erkannt werden. Diese Methode in verschiedenen Versionen war in der Vergangenheit die am häufigsten verwendete bei solchen festverdrahteten Triggern. Da jedoch im Laufe der Zeit die Mikroprozessoren immer schneller und leistungsfähiger wurden, ist es heute möglich, einfache Berechnungen zur Spur-Rekonstruktion in der Größenordnung von einigen zehn Mikrosekunden auszuführen. Insgesamt benötigen solche Programme jedoch noch heute Rechenzeit in der Größenordnung von einigen Millisekunden. Da jedoch moderne Detektoren einige 100nsec nach einem relevanten Ereignis ein Triggersignal für die Kalorimeter (Gate für Energiesummen usw.) benötigen, sind die prozessorgestützten Trigger zu langsam. Diese Trigger können jedoch die Triggerentscheidungen der schnellen Trigger weiter verifizieren und nachträglich korrigieren. Daher steigt der Bedarf an Mikroprozessor gestützten Triggern, welche als zusätzliche Trigger eingesetzt werden.

1.3. Inhalt der Arbeit

In dieser Arbeit wird die Notwendigkeit und die Möglichkeit für den Einsatz eines Mikroprozessorssystemes als zusätzliche Triggerkomponente bei dem Experiment H1 untersucht. Dieses Mikroprozessorsystem soll die Spuren geladener Teilchen in der zentralen Driftkammer von H1 rekonstruieren und den Transversalimpuls und die Multiplizität dieser Spuren für eine Triggerentscheidung verwenden. Um flexibel den noch weitgehend unbekannten Untergrund des Speicherringes HERA zu berücksichtigen, muß dieses Triggersystem frei programmierbar sein. Da es in das bestehende Triggerkonzept von H1 eingegliedert werden soll, muß ein Triggersignal nach maximal 0,5msec geliefert werden.

Der Name des Systems soll MONIKA II

(*Ein* **M***ikroprozessor* **On***li***n***e Trac***k* **A***nalysator*)

sein.

Das zweite Kapitel dieser Arbeit beschreibt die Besonderheiten der zentralen Driftkammer des Detektors H1. Die Signale dieser Driftkammer dienen dem untersuchten Trigger als Daten. Das Kapitel 3 beschreibt das Triggerkonzept von H1 und untersucht die Notwendigkeit eines zusätzlichen Spurtriggers. In Kapitel 4 werden die untersuchten Triggeralgorithmen vorgestellt und und miteinander verglichen.

Als kurze Übersicht stellt das Kapitel 5 dann drei verschieden Prozessorkonzepte dar. Ausfühlicher werden diese Prozessorkonzepte in entsprechenden Anhängen aufgezeigt. Kapitel 6 stellt das vollständige Triggerprogramm dar, welches in Kapitel 7 mit Monte–Carlo–Daten in seinen Eigenschaften ausgetestet wurde. Die Kosten eines solchen Triggersystemes werden in Kapitel 8 zusammengestellt und in Kapitel 9 Erweiterungen und Verbesserungen des vorgestellten Triggersystems erörtert.

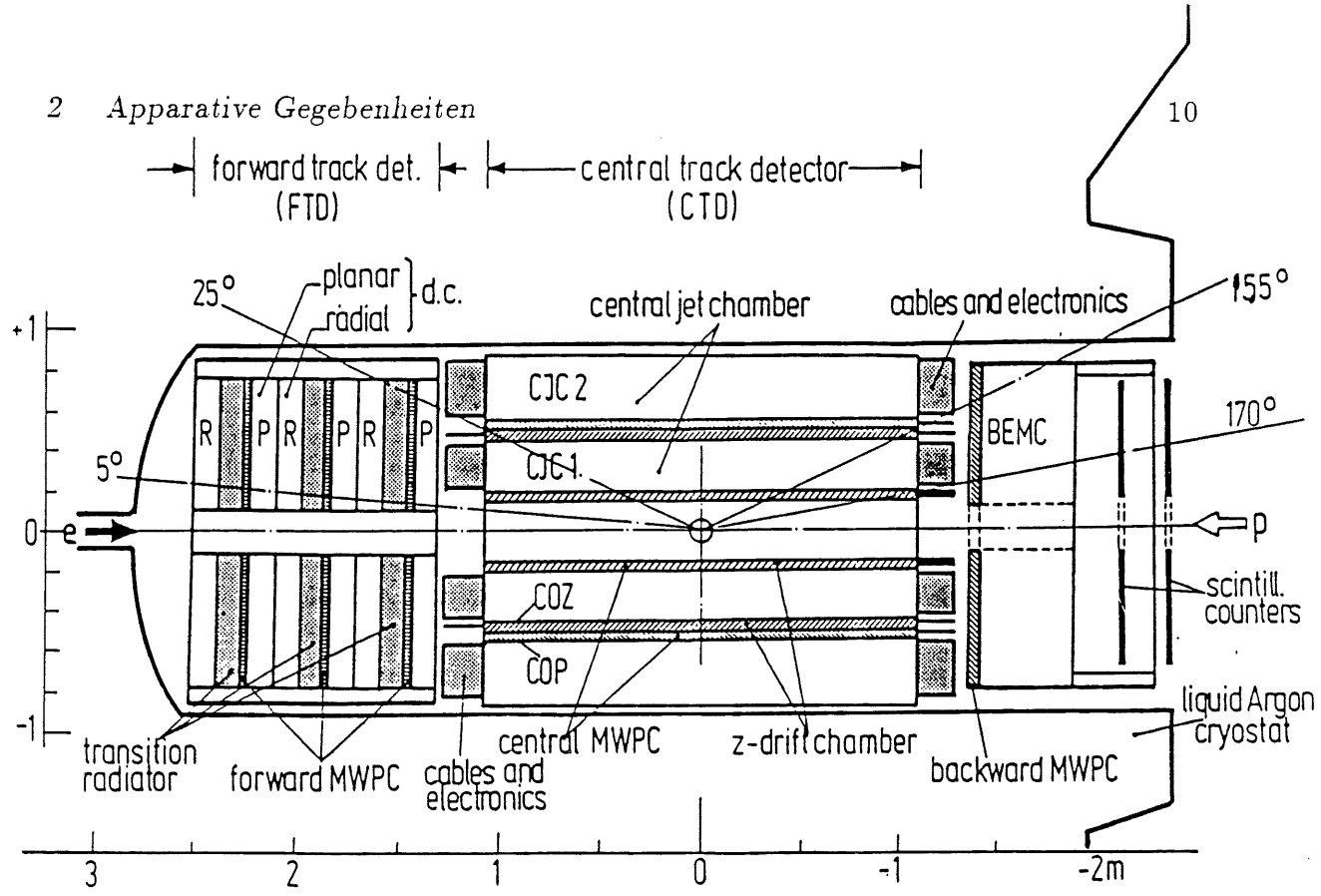


Abbildung 3: Skizze der Driftkammern.

2 Apparative Gegebenheiten

2.1. Die zentrale Driftkammer

Die zentrale Jet–Driftkammer[1] (englisch: central jet chamber, CJC) ermöglicht für geladene Teilchen eine Bestimmung der Koordinaten senkrecht zur Strahlachse (R–Φ–Koordinaten) und in der in der Strahlachse (z–Koordinate) (über Ladungsteilung). Die Ortsauflösung beträgt $\sigma_{r\Phi} = 100\mu m$ und $\sigma_z' = 24mm$.

Die CJC nimmt ein zylindrisches Volumen um das Strahlrohr ein. Sie ist aus mechanischen Gründen in zwei unabhängige Segmente radial unterteilt. Jede dieser Segmente ist aus parallel zur Strahlachse gespannten Drähten aufgebaut. Insgesamt deckt die Kammer einen Winkelbereich von 87mRad $< \Theta <$ 3,0Rad ab.

Das innere Segment ist in der Φ–Richtung in 30 und das äußere in 60 "Superzellen" unterteilt. Diese Superzellen bestehen radial aus 24 (inneres Segment) oder 32 (äußeres Segment) Driftstrecken. Die Driftstrecken sind aus zwei begrenzenden Kathodendrähten (englisch: cathode wire) und einem zentralen Meßdraht (englisch: sense wire) aufgebaut. (siehe Abbilung 4). Die Driftstrecken werden von den drüberliegenden und drunterliegenden Driftstrecken durch Potentialdrähte (englisch: potential wire) abgeschirmt. In der Abbildung 4 ist schematisch die CJC dargestellt. Eine Superzelle ist nach rechts herausgezeichnet worden. In dieser herausgezeichneten Superzelle sind für einige Driftstrecken die elektrischen Feldlinien und die Isochronen der driftenden Elektronen eingezeichnet. In dieser Abbildung

[1] siehe hierzu auch Literatur [2]

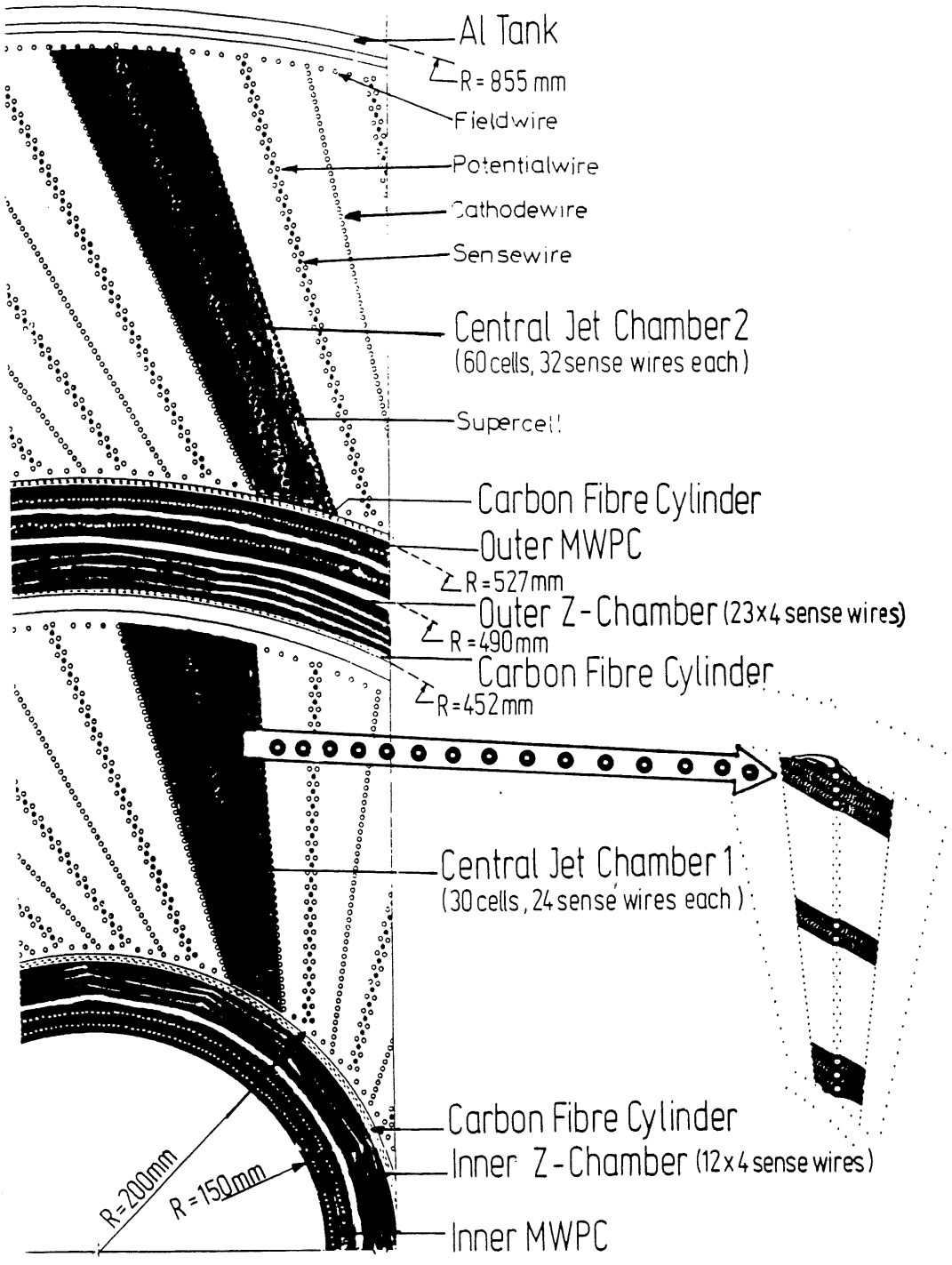


Abbildung 4: Zellstruktur der zentralen Driftkammer.

ist auch zu erkennen, das die Driftstrecken um 30° gegenüber der radialen Richtung geneigt sind, um den Lorentzwinkel auszugleichen. Die Signale der Meßdrähte werden mit einigen 100MHz ausgelesen. Da die Elektronen eine Driftgeschwindigkeit von $3.64 \times 10^4 \frac{m}{sec}$ in dem gewählten Gas (hauptsächlich Argon) besitzen, ergibt sich eine Ortsauflösung innerhalb der Driftstrecken von ca. $100 \mu m$ [3].

Eine supraleitende Magnetspule (siehe Abbildung 2) erzeugt im gesamten zentralen Detektor ein homogenes Magnetfeld von $1{,}20 \pm 0{,}02 Tesla$ parallel zur Strahlachse. Somit krümmen sich die Spuren geladener Teilchen in der R–Φ–Ebene, wodurch die Impulse der Teilchen bestimmt werden können.

2.2. Die Datenauslese der zentralen Driftkammer

Geladene Teilchen ionisieren das Gasgemisch der CJC. Die freiwerdenden Elektronen lösen dort in den Meßdrähten einen Ladungsfluß aus. Dieser Ladungsfluß wird an beiden Enden des Meßdrahtes gemessen und mit Analog–Digital–Wandlern mit ca. 100Mhz digitalisiert (siehe Abbildung 5). Da die erzeugten Daten der schnellen Analog–Digital–Wandler nicht innerhalb von z.B. 0,5msec ausgelesen werden können, sind diese für Trigger nicht verwendbar. Daher werden zusätzlich die Signale der beiden Meßdrahtenden analog addiert und durch Synchronisatoren alle 96nsec (zeitlicher Abstand der Teilchenpakete bei dem Speicherring HERA, welcher als "bunch–clock" den Experimenten zur Verfügung gestellt wird) zu Diskriminatoren geleitet. Diese Diskriminatoren prüfen das Signal auf die Überschreitung einer Mindestschwelle. Das Ergebnis dieser Prüfungen (Ja = 1, Nein = 0) werden in Schieberegistern zwischengespeichert. Die Schieberegister besitzen die Eigenschaft, erst alle vorhergehenden Daten um eine Stelle (in der Abbildung 5 um einen Kasten nach unten) zu verschieben und dann den neuen Wert zu speichern. Der in der Abbildung unterste Wert geht dabei verloren. Der Inhalt dieser Schieberegister (das Ergebnis der letzten 16 Überprüfungen) steht jederzeit parallel den Triggern als Datenquelle zur Verfügung. Bei der erwähnten Driftgeschwindigkeit der Elektronen und der Meßdauer von 96nsec ergibt sich eine Ortsauflösung in der Φ–Richtung von 3,5mm. Der radiale Abstand der Driftstrecken von 7,9mm bedingt eine Auflösung von eben diesen 7,9mm in der R–Richtung. Eine Auflösung in Z–Richtung ist nicht vorgesehen.

Geladene Teilchen erzeugen jedoch nicht in jeder Driftstrecke ein Signal. Dies hat folgende Gründe:

1. Ausfall von Analog–Digital–Wandlern, Addierer, usw.
2. Mechanische Fehler welche zu Inhomogenitäten im elektrischen Feld der Driftstrecken führen.
3. Mechanische Fehler und Verunreinigungen, welche eine Verringerung der angelegten Hochspannung an den Meß– und Kathodendrähten zur Folge haben und somit die Gasverstärkung herabsetzen.

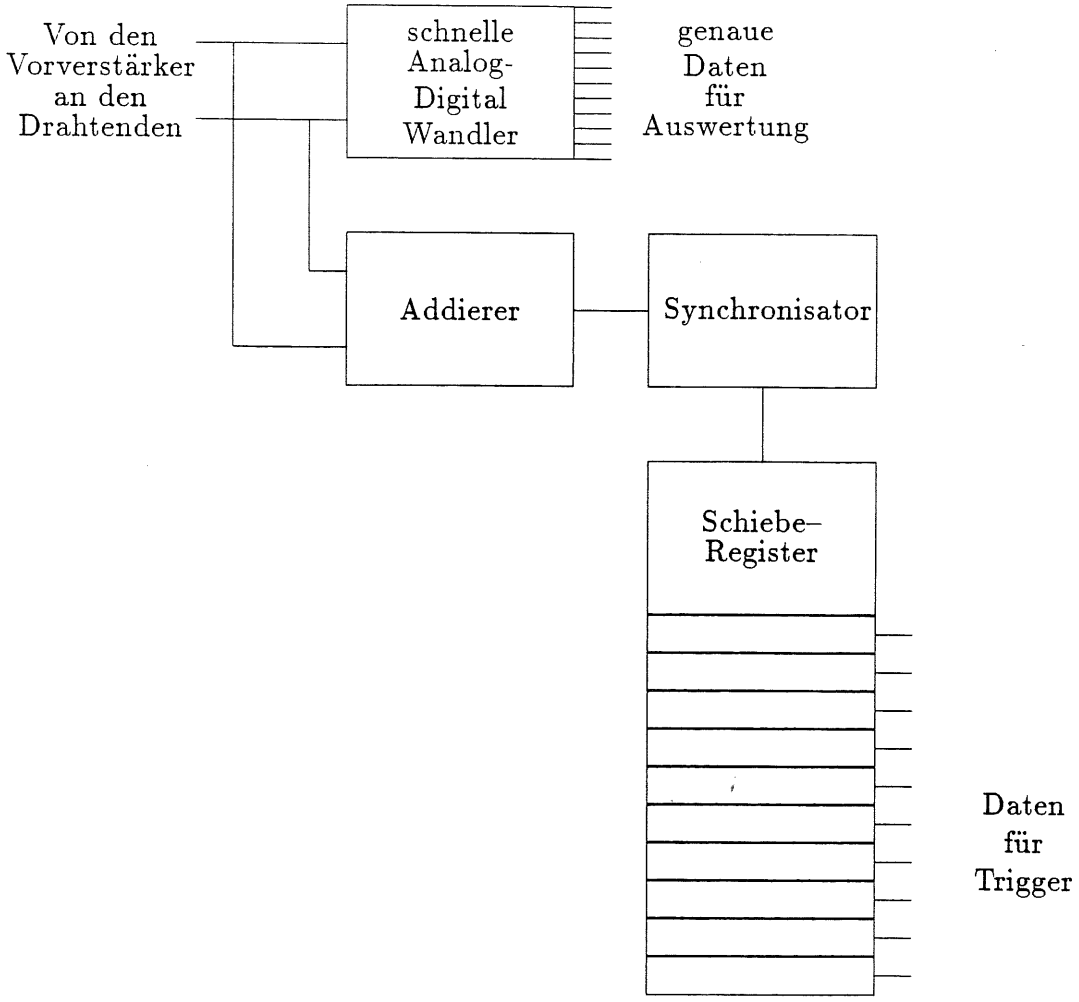


Abbildung 5: Datenauslese der CJC

4. Die Produkte einer Gas—Ionisation durch ein geladenes Teilchen schirmen das elektrische Feld für ein zeitlich oder örtlich dicht folgendes Teilchens ab.

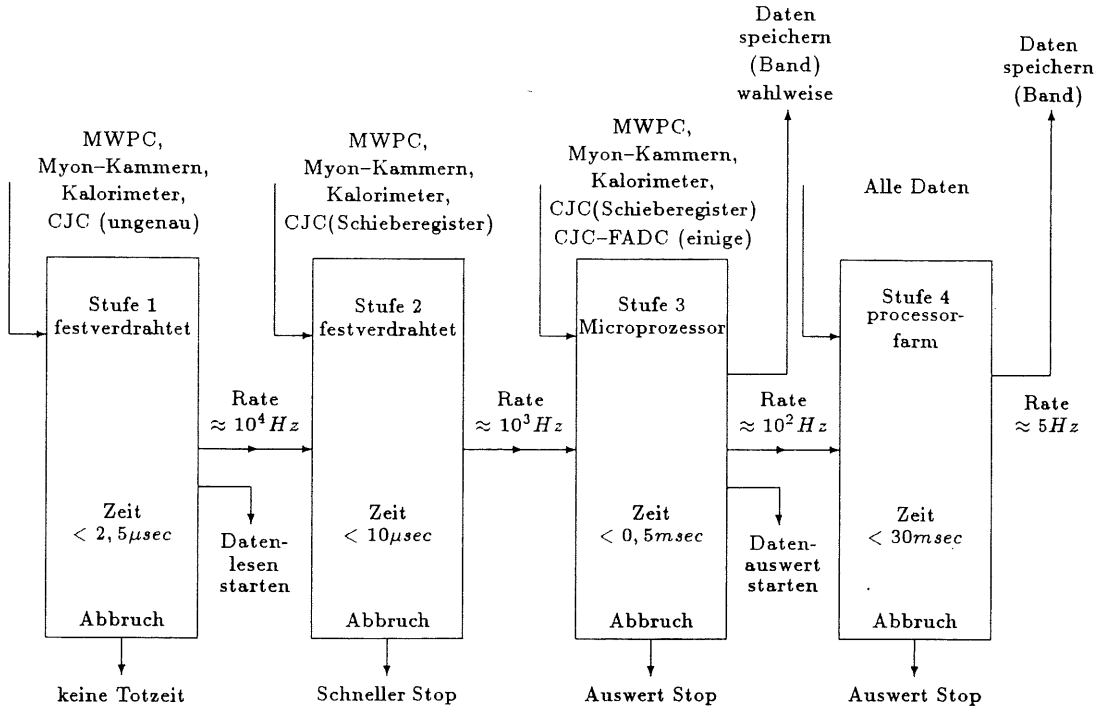


Abbildung 6: Triggersystem

3 Trigger bei H1

3.1. Die Triggerstufen

Heutige Triggersysteme bestehen nicht mehr aus einer Komponente, sondern aus verschiedenen Triggerstufen. Diese benutzen für die Triggerentscheidung die zum Zeitpunkt der Entscheidung verfügbaren Daten. Somit müssen Trigger höherer Stufe eine größere Datenmenge verarbeiten als Trigger niederer Stufen. In Abbildung 6 sind die verschiedenen Triggerstufen, welche bei H1 geplant sind, schematisch dargestellt. Oberhalb der Triggerstufen sind in der Abbildung die Quellen der Daten für die Trigger aufgeführt. Trigger bis zur Stufe 3 benutzen nur Daten spezieller Detektorkomponenten (z.B. Kalorimeterdaten). Erst in der Stufe vier finden die Daten aller Detektorkomponenten Verwendung. Die Triggerstufen unterscheiden sich durch folgende Eigenschaften:

1.Stufe: Diese Trigger bestehen aus festverdrahteten, parallelarbeitenden Logikschaltungen, welche spezielle Triggerdaten (z.B. der Inhalt der Schieberegister der Abbildung 5) verwenden. Da der Preis und der Platzbedarf dieser Trigger

stark von der gewählten Auflösung abhängt, wird die Auflösung der Triggerdaten verkleinert. Dies geschieht z.B. durch logisches Zusammenfassen der in Abbildung 5 parallel gezeichneten Ausgänge der Schieberegister (entspricht der boolschen "*oder*" Verknüpfung). Das Auslesen der Daten des gesamten Detektors H1, welche durch die Entscheidung dieser Triggerstufe eingeleitet wird (siehe Abbildung 8), benötigt ca. 0,8msec. Während dieser gesamten Datenauslese kann der Detektor keine neuen Signale messen. Sie stellt somit die Totzeit des Detektors dar. Bei dem Detektor H1 wird eine mittlere Totzeit von 10% angestrebt. Da die Trigger der ersten Stufe diese Totzeit einleiten, und die Trigger der zweiten Stufe diese Totzeit erst nach maximal $10\mu sec$ abbrechen, sollte die Rate der Ereignisse welche den ersten Trigger passieren nicht größer als 10^4Hz $\left(\frac{1\text{sec}-90\%}{10\mu sec} = 10^4\right)$ sein. Nach 2,5μsec benötigen die Kalorimeter das Signal dieser Triggerstufe, um die Summenbildung der deponierten Energien in den verschiedenen Kalorimeterstapeln (Tower) zu unterbrechen.

2.Stufe: Die 2. Stufe hat nach 10μsec ein Triggersignal zu erzeugen. Dies ermöglicht Triggerkonzepte, welche die Daten nicht zeitlich parallel sondern zeitlich hintereinander (sequenziell) verarbeiten (spezielle sequenziell arbeitende Spezialprozessoren). Ein Trigger von Behrend, DESY–Hamburg, welche die Daten der Vorwärts– Spurenkammern verwenden, ist jedoch im Moment das einzige Konzept eines Triggers dieser Stufe. Da die Trigger der Stufe Drei im Mittel nach $100\mu sec$ die Totzeit des Detektors unterbrechen, sollten nur ca. 10^3 Ereignisse pro Sekunde $\left(\frac{1\text{sec}-90\%}{0{,}1\text{msec}} = 10^3\right)$ diesen Trigger passieren.

3.Stufe: Dieser Triggerstufe stehen die Triggerdaten und einige genaue Daten z.B. der schnellen Analog–Digital–Wandler der CJC (siehe Abbildung 6) zur Verfügung. Da die Trigger dieser Stufe das Auslesen der Daten nach Möglichkeit noch unterbrechen soll, muß in maximal 0,8 msec ein Triggersignal an den *event coordinator* geliefert werden.

Die zur Verfügung stehende Zeit reicht aus, um ein Mikroprozessorsystem einzusetzen, welches die Triggerdaten mit einer höheren Genauigkeit weiterverarbeiten kann als die Triggerstufen Eins und Zwei. Solche Mikroprozessorsysteme zeichnen sich aus durch hohe Flexibilität und niedrige Entwicklungskosten und sind somit in der Inbetriebnahmephase des Detektors geeignet, erst einmal die auftretenden Untergrundereignisse zu studieren und so die Konzepte der ersten und zweiten Triggerstufe zu verbessern. Da das Signal dieser Triggerstufe die vollständige Datenauslese des Detektors, was eine Totzeit von 0,8msec zur Folge hat, bewirkt, darf die den Trigger passierende Ereignisrate nicht höher als 100Hz $\left(\frac{1\text{sec}-90\%}{0{,}8\text{msec}} \approx 100\right)$ sein.

4.Stufe: Diesem Trigger stehen alle Daten des Detektors (alle Komponenten wie z.B. die Daten der Analog–Digital–Wandler und Schieberegister in der Abbildung 5) zur Verfügung. Er benutzt die Daten des Ereignisspeichers. Dieser

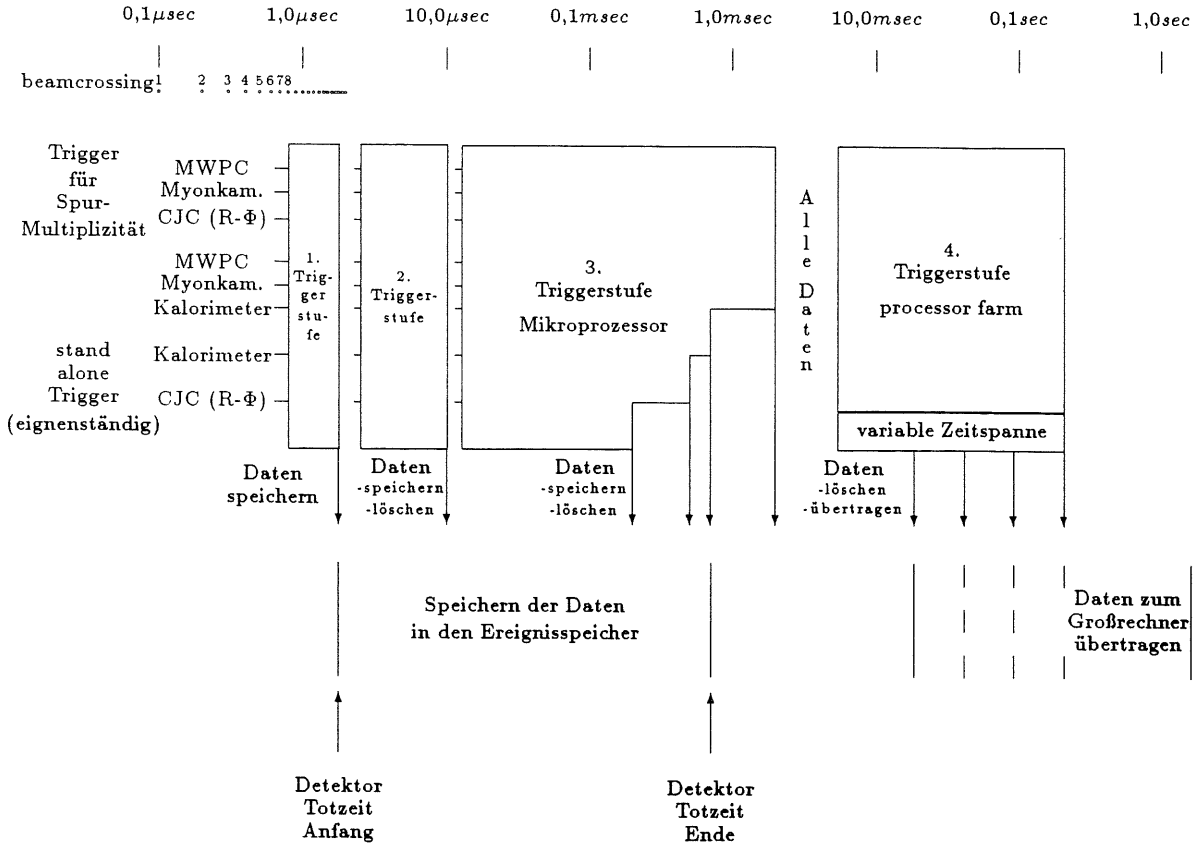


Abbildung 7: Zeitdiagramm der verschiedenen Triggerstufen

Trigger verhindert nicht mehr die Datennahme eines Ereignisses in den Ereignisspeicher, sondern verhindert die Speicherung der Daten auf Datenbändern. Diese Triggerstufe hat somit keinen Einfluß auf die Totzeit des Detektors, sondern reduziert die Ereignisrate auf die maximale Ereignisspeicherrate des Experimentes H1 (5Hz). Nach 30msec sollte dieser Trigger sein Triggersignal an den *event coordinator* liefern. Der Trigger besteht aus einem Rechnerverbund (englisch: processor farm) von vielen gleichwertigen Rechnern.

In Abbildung 7 ist der zeitliche Rahmen der oben genannten Triggerstufen graphisch dargestellt. Es ist zu beachten, daß eine logarithmische Zeiteinteilung in der Abbildung gewählt wurde. In Abbildung 8 ist der gesamte Datenfluß des Detektors H1 schematisch dargestellt. Diese Abbildung kann logisch unterteilt werden in zwei Hälften. Die linke Hälfte zeigt den Hauptdatenfluß des Detektors H1 auf. Gesteuert wird dieser Datenfluß von dem "*event coordinator*". Dieser Coordinator erhält seine Steuerinformation von dem Triggersystem, welches in der rechten Hälfte der Abbildung 8 dargestellt ist. Die von den einzelnen Triggerstufen kommenden Signale und deren Wirkung sind als Pfeile angedeutet. Die Systemkontrolle dient

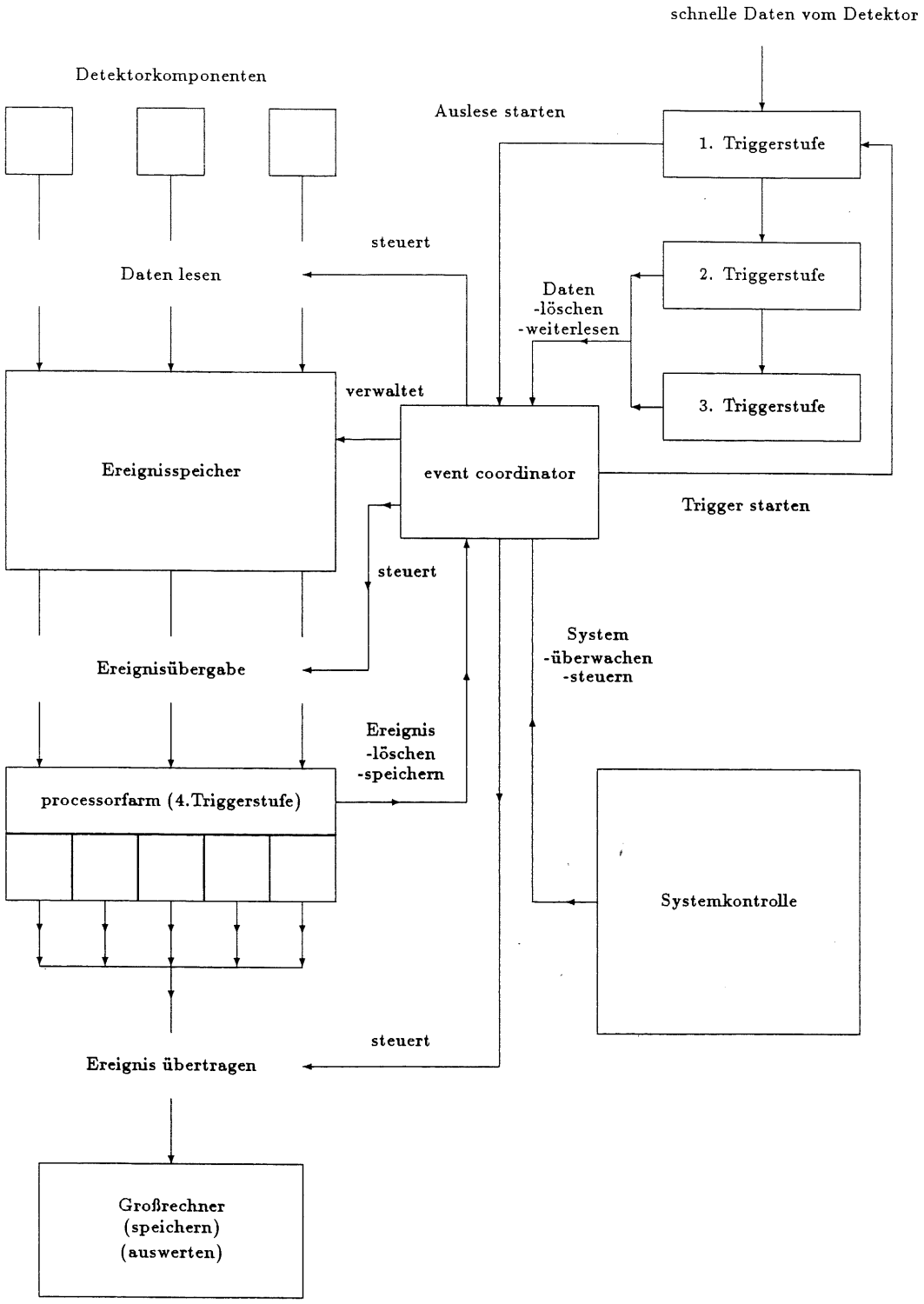


Abbildung 8: Datenfluß bei H1

zur Hauptsache zur Inbetriebnahme, Überwachung der Funktionsfähigkeit und zur Außerbetriebnahme des gesamten Detektors.

3.2. Die geplanten Triggerkomponenten bei H1

Bei H1 sind folgende Trigger für spezifische Detektorkomponenten geplant:

Kalorimetertrigger: Diese Trigger werten die deponierte Energie im Vorwärts-, Rückwärts- und zentralen Bereich des Kalorimeters und den Zeitpunkt der Wechselwirkung aus und generieren daraus die Triggersignale (englisch: high energie trigger).

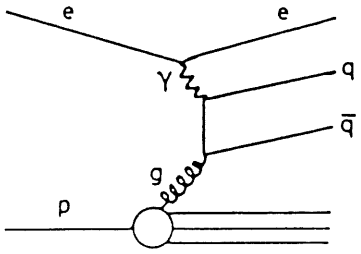
Myontrigger: Um alle Ereignisse bei denen Myonen entstehen zu triggern, wird aus den Signalen der Myonkammern ein weiteres Triggersignal gebildet (englisch: event class trigger). Dieser Trigger ist notwendig, da solche Ereignisse häufig in den Kalorimetern eine Energie unterhalb der Triggerschwellen deponieren.

Z–Vertex–Trigger: Diese Trigger bestimmen aus den Z–Kammersignalen den Wechselwirkungspunkt eines Ereignisses in der Z–Richtung. So können sie wirkungsvoll die Kalorimetertrigger unterstützen (englisch: central tracking trigger).

MWPC–Trigger: Diese Trigger benutzen die Vorwärts–Driftkammern, um den Wechselwirkungspunkt eines Ereignisses, welches einen hohen Impuls in Vorwärtsrichtung hat, in der Z–Richtung zu bestimmen (englisch: forwards tracking trigger).

3.3. Notwendigkeit eines Driftkammertriggers

Die Haupttrigger des Experimentes H1 sind die Kalorimetertrigger. Diese Trigger sind jedoch unzuverlässig, falls physikalisch relevante Ereignisse zu wenig Energie in den Kalorimetersegmenten deponieren. Daher müssen Trigger, welche die bei einem Ereignis entstehenden geladenen Teilchen und deren Impuse zur Generierung des Triggersignals verwenden, die Kalorimetertrigger unterstützen. Die bisher geplanten Trigger für geladene Teilchen arbeiten nur in der R–Z–Ebene und haben im zentralen Detektorbereich eine schlechte Auflösung (in der Größenordnung von 10^{-2}m) auf Grund der kleinen Anzahl von Z–Kammern (5 Lagen im zentralen Bereich, zwei direkt am Strahlrohr und drei zwischen den beiden Segmenten der CJC). Ist es sinnvoll, außer den oben erwähnten Triggern, noch zusätzliche Driftkammertrigger zu entwickeln? Diese Frage ist äquivalent zu der Frage, ob es Ereignisse gibt, welche eine Energiedeposition in den Kalorimetern von $E < 10$–30GeV (voraussichtliche Triggerschwellen der Kalorimeter) und eine hohe Multiplizität geladener Ereignisprodukte in den Driftkammern aufweisen. Zu dieser Frage gibt es Unter-



Feynman Diagramm
$\gamma g \longrightarrow c\bar{c}, b\bar{b}, t\bar{t}$

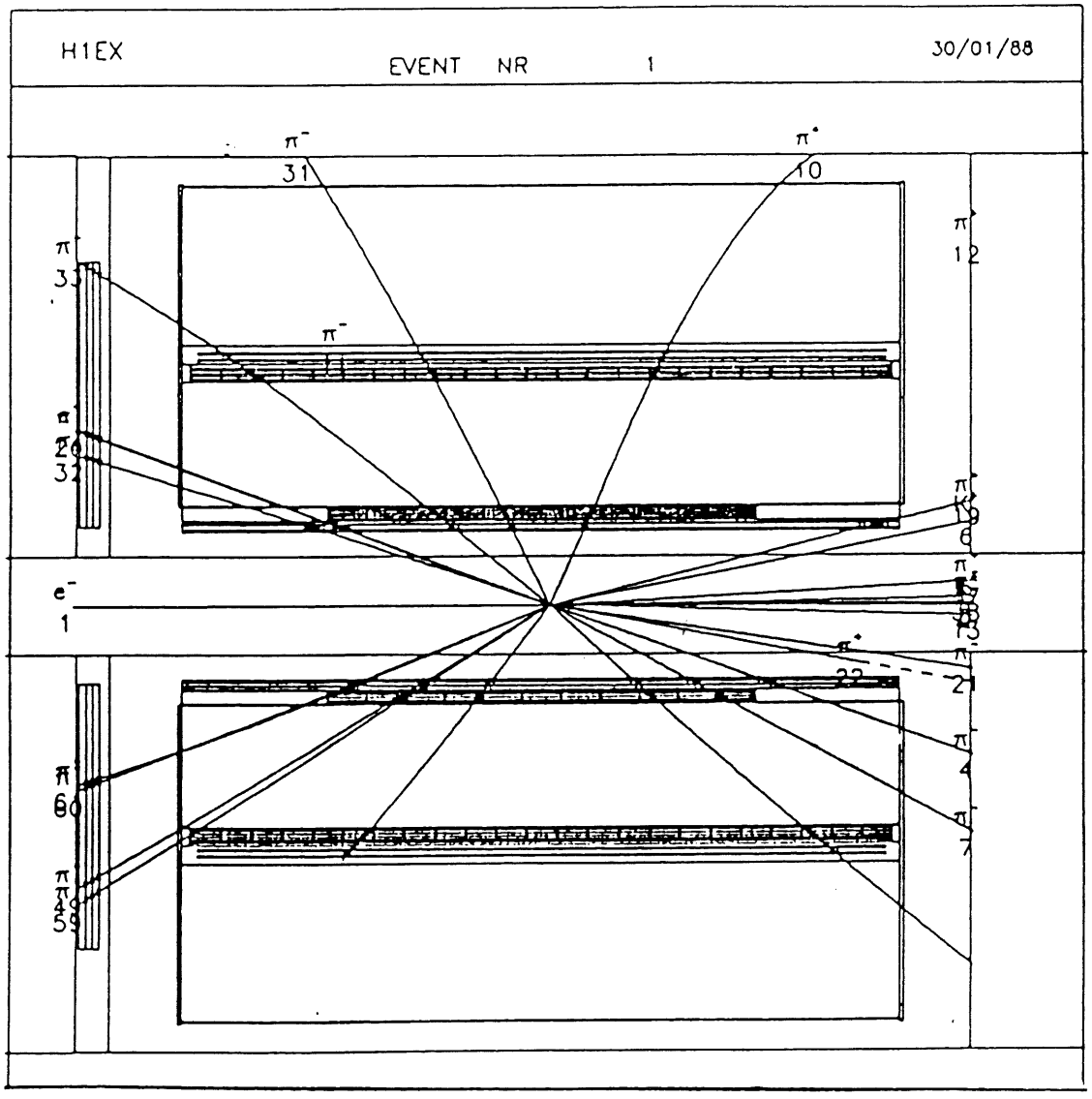


Abbildung 9: R–Z Projektion eines $\gamma g \rightarrow b\bar{b}$ Ereignisses

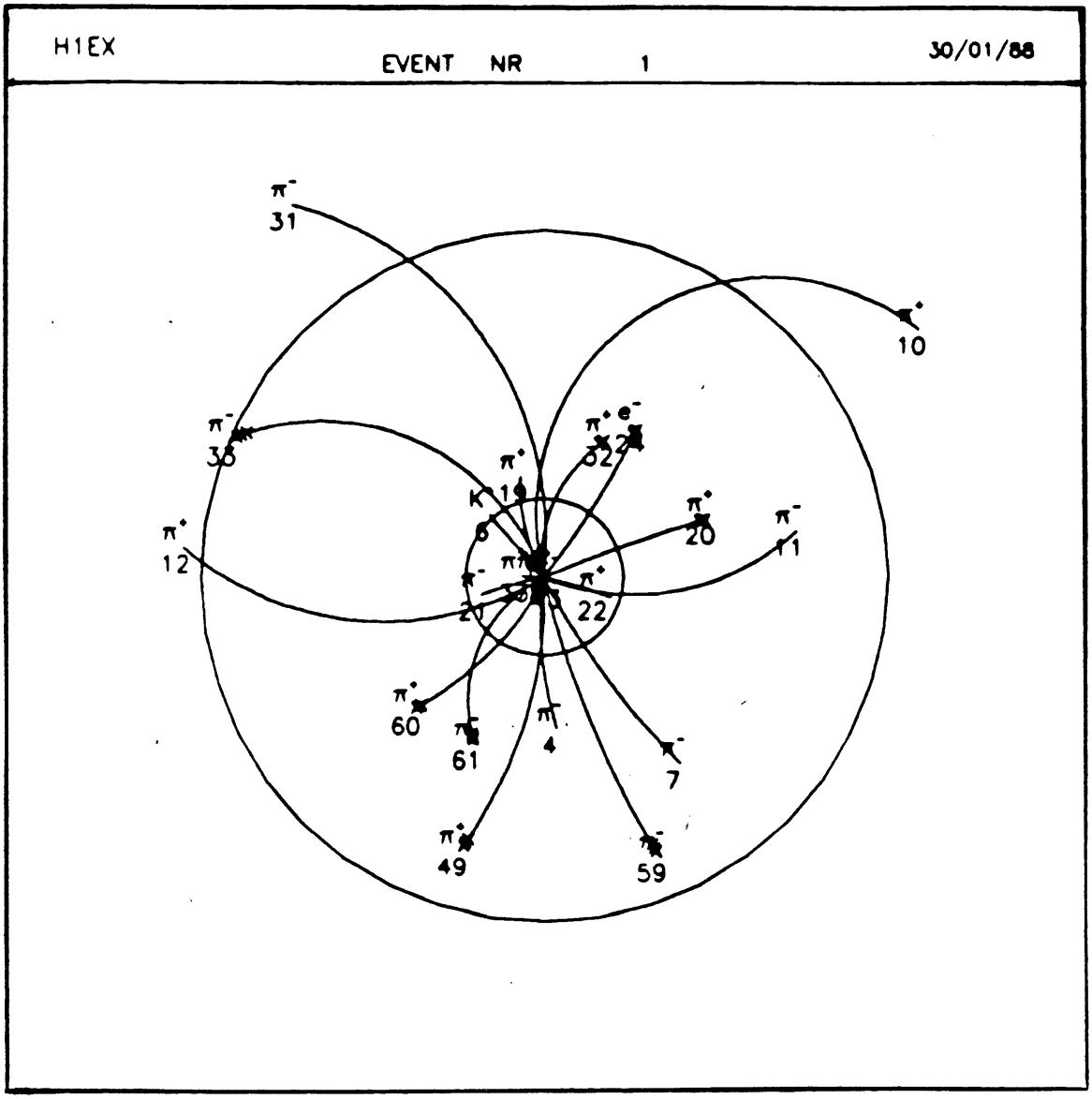


Abbildung 10: R–Φ Projektion eines $\gamma g \rightarrow b\bar{b}$ Ereignisses

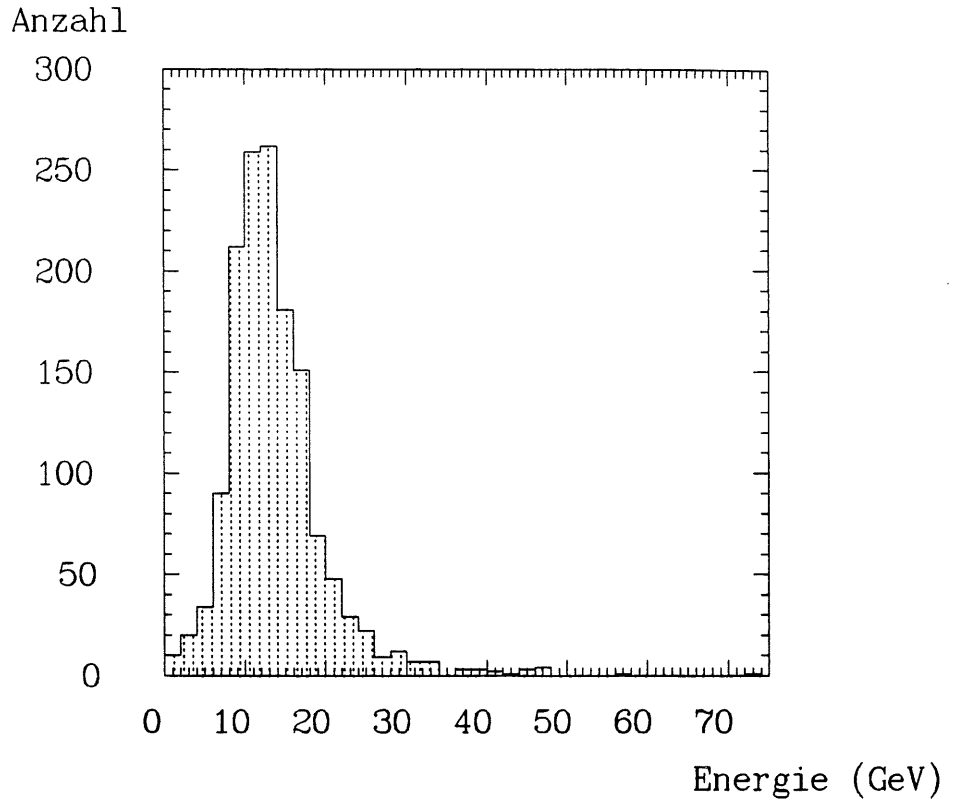


Abbildung 11: Die Transversalenergie aller geladenen Teilchen mit $\Theta > 5°$

suchungen von Th. Naumann [40], D.H. Saxon [41], H.-J.Behrend [42] und F. Eisele [43]. Sie zeigen, daß die Energie der Ereignisprodukte der γ–Gluon–Fusion (siehe Abbildung 9 und 10) häufig unter 10–30GeV liegen Die Ereignisprodukte der Reaktion $\gamma g \rightarrow b\bar{b}$ weisen nach [40] nur in 12%–70% der Fälle eine transversale Energie von $E_T > 10$–30GeV (siehe Abbildung 11 und 12) auf.

Um diese Ereignisse, welche von den Kalorimetertriggern nicht getriggert werden können, mit Driftkammertriggern zu triggern, ist es Notwendig, daß die Multiplizität der geladenen Teilchen in der CJC hoch genug ist. Die mittlere Multiplizität der geladenen Ereignisprodukte des gesamten Ereignisses ist nach [40] 24. H.-J. Behrend zeigte in Abbildung 13 daß 85% der simulierten $\gamma g \rightarrow b\bar{b}$ Ereignisse mindestens 3 geladene Teilchen im zentralen Bereich emittieren. (F.Naumann beobachtete bei 72% der Ereignisse mindestens 3 geladene Teilchen im Winkelbereich von $25° < \Theta < 155°$; siehe Abbildung 14). Solche Ereignisse können durch Spurtrigger der zentralen Driftkammer getriggert werden.

Wie könnte das Triggerkonzept für H1 aussehen? Dazu gäbe es drei Möglichkeiten:

1. Man verwendet ausschließlich Kalorimetertrigger. Dann muß jedes Ereignis

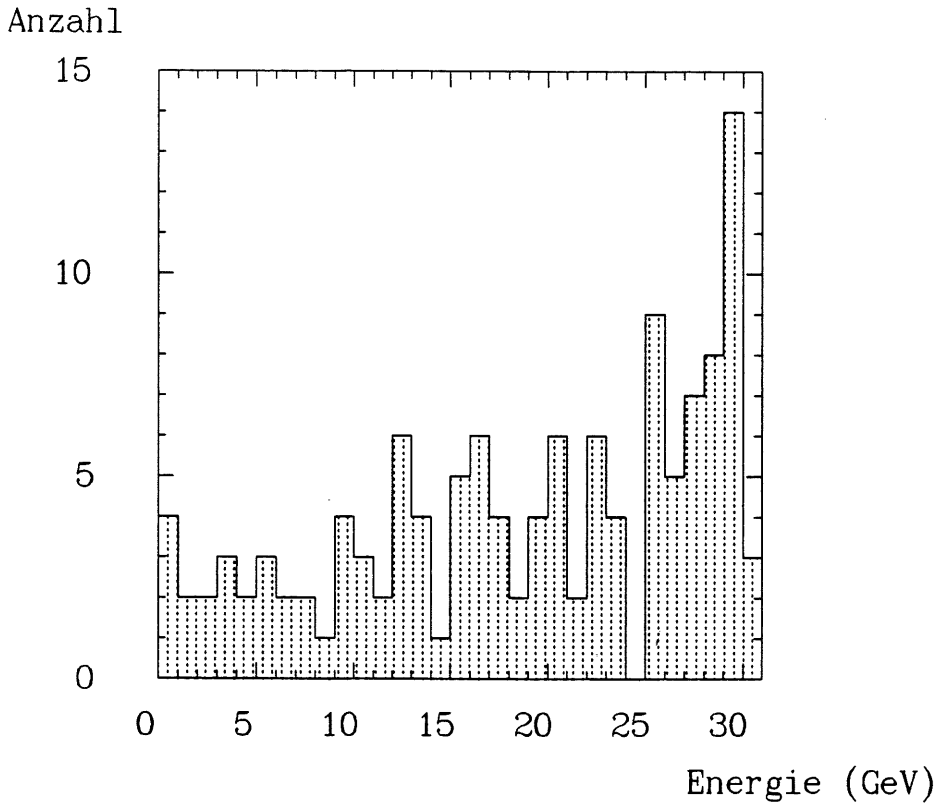


Abbildung 12: Die Transversalenergie des gestreuten Elektrons

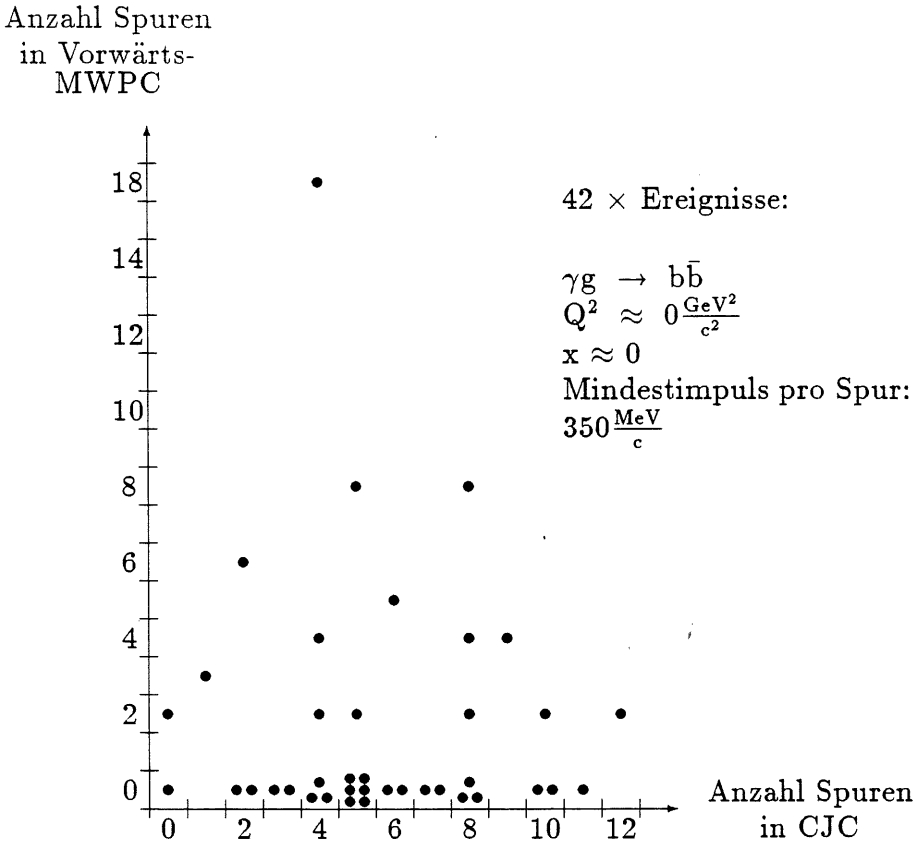


Abbildung 13: Spurmultiplizität in der CJC und MWPC

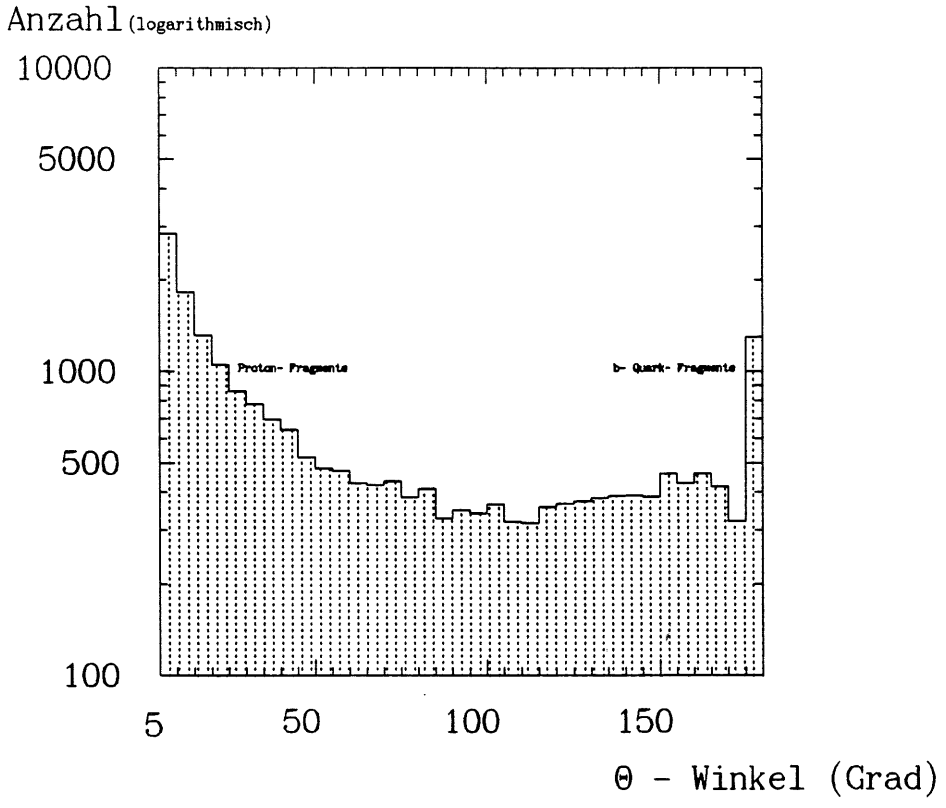


Abbildung 14: Die Verteilung aller geladenen Teilchen nach der Fragmentation

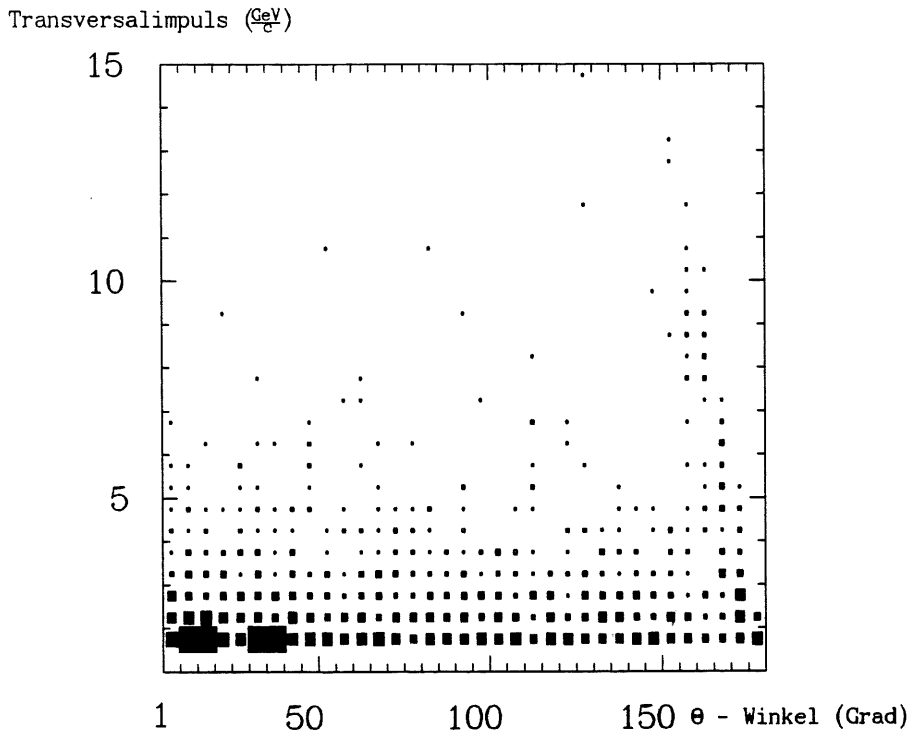


Abbildung 15: Der Transversalimpuls aller geladenen Teilchen nach der Fragmentation

eine deponierte Energie im Kalorimeter von $E_{deponiert} > E_0$ aufweisen.

2. Man verwendet ausschließlich einen Driftkammertrigger. Dann muß die Multiplizität der geladenen Teilchen ($N_{charged}$) in den Driftkammern mindestens $N_{charged} > N_0$ mit $P_T(N) > P_{T\ min}$ betragen.

3. Es werden beide Triggersysteme eingesetzt, wodurch die Ereignisse nun den beiden Bedingungen $N_{charged} > N_1$ und $E_{deponiert} > E_1$ genügen müssen. Untersuchungen von H.-J. Behrend zeigten, daß jedoch bei gleicher Anzahl der Ereignisse, welche die Trigger passieren, gilt: $N_1 < N_0$ und $E_0 < E_1$.

Es ist zu erwarten, daß Spurtrigger (Driftkammertrigger) in Zusammenarbeit mit den Kalorimetertriggern die Effizienz des H1–Triggerkonzeptes wirkungsvoll unterstützen können.

3.4. Die Untergrundereignisse

Vier Arten des Untergrundes werden bei HERA– Experimenten erwartet (siehe Tabelle 3):

- Wechselwirkung des Protonenstrahles mit Rest– Gas– Molekülen (englisch: beam–gas events). Bei diesen Wechselwirkungen werden hauptsächlich energiereiche Teilchen in Protonenrichtung erzeugt (englisch: upstream beam- gas collision,UBG). Die angeregten Kerne emittieren niederenergetische Neutronen bzw. Protonen (englisch: spallation protons, spallation beam gas events, SBG) unter großen Winkeln.

- Protonen, welche nicht mehr fokussiert werden können, treffen das Strahlrohr und erzeugen schnelle hadronische Schauer (englisch: lost protons)

- Elektronen, welche nicht mehr fokussiert werden können und in die Detektoren gestreut werden (englisch: off momentum elektrons).

- Die Intensität der Elektron- Synchrotron- Strahlung ionisiert das Gas in der CJC (im Mittel 10 Signale pro Kreuzung des Protonen– und Elektronenpaketes).

Die auf Grund der Ereignisrate von $2,0 \times 10^5 Hz$ (bei Beachtung aller Ereignisse bis 100m vor dem H1 Detektors) am schwierigsten zu reduzierenden Untergrundereignisse sind die '*lost protons*' Ereignisse. In der Praxis ist zu hoffen, daß viele dieser *lost protons* und deren Zerfallsprodukte durch Blenden im Strahlrohr von den Detektoren ferngehalten werden. Die *lost proton*- Ereignisse im Bereich der Detektoren erzeugen hadronische Schauer in dem Vorwärtskalorimeter und niederenergetische Zerfallsprodukte in den zentralen Detektorkomponenten.

Die '*off momentum elektrons*' haben ebenfalls eine hohe Rate (5×10^3 bei Beachtung aller Ereignisse bis 5m vor dem Detektor H1), werden jedoch vor allen Signale

in dem rückwärtigen Elektromagnetischen–Kalorimeter erzeugen. Diese Ereignisklasse bereiten Spur–Triggern (englisch: charged trigger) im zentralen Bereich keine Probleme, da die Multiplizität der Ereignisse (Zahl der geladenen Teilchen) und die Impulse der Teilchen in den zentralen Detektorkomponenten sehr klein sind.

Schwieriger ist die Situation bei den SBG's, UBG's und den lost Protons. Diese Ereignisse haben zum Teil eine hohe Multiplizität in der zentralen Driftkammer. Da jedoch die Energie und der Impuls [42] dieser Teilchen gering ist, können sie von Kalorimeter– und Driftkammertriggern unterdrückt werden. Die Rate der SBG's (auch der UBG's) ist proportional zum Druck im Strahlrohr. Bei Inbetriebnahme und der Testphase des Beschleunigers und der Detektoren ist daher mit einer hohen Rate dieser Ereignisse zu rechnen. Da lost Proton Ereignisse und UBG's das gleiche Erscheinungsbild haben und im Allgemeinen damit gerechnet wird, daß sie den größten Anteil des Untergrundes bilden, wurden solche Ereignisse gewählt, um die Rate der Untergrundsereignisse zu bestimmen, welche der in dieser Arbeit vorgestellte Trigger nicht unterdrücken kann.

Da bis zum heutigen Tage keine Daten für Untergrundereignisse bei 800GeV vorliegen, ist die Bestimmung der Untergrundraten schwierig. Je nach Bestimmungsart der Rate, nach theoretischen Modellen oder Hochrechnung der Beobachtungen an bestehenden Beschleunigern, ergeben sich große Unterschiede. Ich bevorzuge in dieser Arbeit die Daten in [33], welche zur Hauptsache aus hochgerechneten Beobachtungen am Speicherring SPS entstanden sind.

In Tabelle 2 und 3 sind einige Ereignisraten von verschiedenen Ereignisklassen aufgeführt. Diese Zahlen beziehen sich auf eine Luminosität von $2 \times 10^{31}\text{cm}^{-2}\text{sec}^{-1}$, einem Druck von 10^{-10}torr, 1×10^{11} Protonen und $3,5 \times 10^{10}$ Elektronen im '*bunch*' und einer '*bunch crossing*' Frequenz 10^7Hz. Für die UBG's wurden Ereignisorte bis 100m vor dem Wechselwirkungspunkt gewählt.

Prozeß	Bedingungen	Rate (Hz)	Rate pro bunch crossing	Referenz
NC	$Q^2 > 3(\mathrm{GeV/c})^2$	3	3×10^{-7}	[6]
	$Q^2 > 5000(\mathrm{GeV/c})^2$	10^{-4}	1×10^{-11}	[6]
CC	alle Q^2	3×10^{-3}	3×10^{-10}	[6]
	$Q^2 > 5000(\mathrm{GeV/c})^2$	5×10^{-4}	5×10^{-11}	[6]
Photo-production	alle	$\approx 10^3$	10^{-4}	[6]
	sichtbare	$\approx 10^2$	10^{-5}	[6]
	$E_{Jet} > 10\mathrm{GeV}$, $\theta_{Jet} > 5^o$	1	10^{-7}	[6]
$ep \rightarrow epe^+e^-$	$W_{e^+e^-} > 1\frac{\mathrm{GeV}}{\mathrm{c}}$			
	$\theta_{e^+e^-} > 2,6^o$	0,14	$1,4 \times 10^{-8}$	[33]
	$\theta_{e^+e^-} > 30^o$	$8,0 \times 10^{-3}$	$8,0 \times 10^{-10}$	[33]
$ep \rightarrow ep\rho$	$8^o < \theta_{e,\gamma} < 172^0$	$4,3 \times 10^{-3}$	$4,3 \times 10^{-10}$	[37]
$ep \rightarrow ep\rho$	$5^o < \theta_{\pi^+\pi^-} < 175^0$	56	$5,6 \times 10^{-6}$	[37]
$\rho \rightarrow \pi^+\pi^-$	$30^o < \theta_{\pi^+\pi^-} < 150^0$	16	$1,6 \times 10^{-6}$	[37]
$ep \rightarrow ep\rho$	$5^o < \theta_{e^+e^-} < 175^0$	$2,6 \times 10^{-3}$	$2,6 \times 10^{-10}$	[37]
$\rho \rightarrow e^+e^-$	$30^o < \theta_{\pi^+\pi^-} < 150^0$	$7,3 \times 10^{-4}$	$7,3 \times 10^{-11}$	[37]
$ep \rightarrow epJ/\Psi$	$5^o < \theta_{e^+e^-} < 175^0$	$1,0 \times 10^{-2}$	$1,0 \times 10^{-9}$	[37]
$J/\Psi \rightarrow e^+e^-$	$30^o < \theta_{\pi^+\pi^-} < 150^0$	3×10^{-3}	3×10^{-10}	[37]
$ep \rightarrow epc\bar{c}$	$45^o < \theta_{e^+e^-} < 135^0$	2×10^{-2}	2×10^{-9}	[38]
$c \rightarrow$ Jet	$P_T^{Jet} > 4\frac{\mathrm{GeV}}{\mathrm{c}}$			
$\bar{c} \rightarrow$ Jet				

Tabelle 2: Raten physikalisch relevanter Ereignisse

Prozeß	Bedingungen	Rate (Hz)	Rate pro bunch crossing	Referenz
Lost Protons	$\tau_{Loss} = 24$ Stunden	$2,0 \times 10^5$	$2,0 \times 10^{-2}$	[33]
UBG (Hadronen)	Sichtbar	$3,3 \times 10^4$	$3,3 \times 10^{-3}$	[38]
	$E_T > 10$ GeV	$2,3 \times 10^2$	$2,3 \times 10^{-5}$	[38]
UBG (Hadronen)	$> 20\sigma$ vom Strahl			
(UA1 1983	r< 30cm	500	5×10^{-5}	[34]
Messungen)	r< 100cm	$1,5 \times 10^3$	$1,5 \times 10^{-4}$	[34]
SBG	$\pm 2,5$m in z	$1,5 \times 10^3$	$1,5 \times 10^{-4}$	[38]
	$\pm$25cm in z	$1,5 \times 10^2$	$1,5 \times 10^{-5}$	[38]
Synchrotron Strahlung (Photonen)	$\pm 2,5$m in z im Strahlrohr	1×10^8	10	[36]
Off Momentum Elektronen	$\pm$5m in z r > 10cm	5×10^3	5×10^{-4}	[35]
Kosmische Strahlung	5,4 m Abschirmung 10×10m (Kalorimeter) 5,4m Abschirmung + Eisen des Kalorimeters	3×10^3	3×10^{-4}	[33]
	4×2m (tracking)	$1,2 \times 10^2$	$1,2 \times 10^{-5}$	[33]
	2×50cm (Vertex Trigger)	0,15	$1,5 \times 10^{-8}$	[33]

Tabelle 3: Raten der Untergrundereignisse

4 Verschiedene Triggeralgorithmen ("tracking"-Algorithmen)

In diesem Kapitel sollen verschiedene mögliche Algorithmen für die Spursuche vorgestellt werden. Da die Ausführungszeit der kritischste Punkt solcher Triggerprogramme ist, wird diese für die verschiedenen Algorithmen abgeschätzt. Heute verwendete Prozessoren haben zum Teil sehr unterschiedliche Eigenschaften. Daher wurden drei Vertreter verschiedener Prozessorklassen für die Zeitabschätzung verwendet.

Alle Algorithmen basieren auf der Geometrie der inneren CJC. Die aus dem Wechselwirkungspunkt kommenden geladenen Teilchen beschreiben infolge des in Strahlrichtung verlaufenden magnetischen Feldes eine Schraubenlinie. Die Projektion einer solchen Schraubenlinie auf eine senkrecht zur Strahlrichtung gelegenen Ebene (R–Φ-Ebene) ist ein Kreisbogen. Die Spur-Rekonstruktion in der R–Φ Ebene beschränkt sich auf die Suche nach den aus dem Zentrum kommenden Kreisbögen.

Alle Algorithmen gehen von einer außenliegenden Driftstrecken (Referenzkammern) aus, da dort die Zahl der Untergrundsignale geringer ist als nahe dem Strahlrohr. Spuren mit einer großen Vorwärts- oder Rückwärtskomponente des Impulses können jedoch dann nicht rekonstruiert werden, da diese meist die zentrale CJC vor der Referenzkammer verlassen. Man könnte diese Fehlerquelle einschränken, wenn die Algorithmen auch Daten der Vorwärtsdriftkammern verwendeten. Diese werden jedoch in keinem der vorgestellten Algorithmen berücksichtigt, es wäre jedoch generell möglich.

Als minimaler Transversalimpuls wurde für alle Algorithmen ein Wert von ca. 300 $\frac{\mathrm{MeV}}{\mathrm{c}}$ gewählt.

4.1. Spursuche unter Verwendung der Kreisgleichung

Dieser Algorithmus basiert darauf, daß ein Kreis durch drei Punkte definiert ist. Mit der Kreisgleichung können dann die weiteren Punkte gesucht werden. Die Punkte der Referenzkammer werden der Reihe nach bearbeitet. Mit jedem Punkt in dieser Kammer wird folgender Ansatz[2] gemacht:

> *Wenn ein Punkt zu einer echten Spur gehört, so muß in einem gewissen Bereich (der vom minimalen Impuls abhängt) in der darunterliegenden Kammer ein weiterer Punkt sein. Läßt sich ein solcher Punkt finden, so kann man annehmen, daß diese beiden Punkte und der Wechselwirkungspunkt auf einem gemeinsamen Kreisbogen liegen. Unter Verwendung der Kreisgleichung können der Radius (R) und der Tangentenwinkel im Wechselwirkungspunkt (Φ_T) berechnet werden. Aus dem Radius und dem Tangentenwinkel berechnet sich der Fortsetzungspunkt in der*

[2] siehe hierzu auch Literatur [7],[8] und [9]

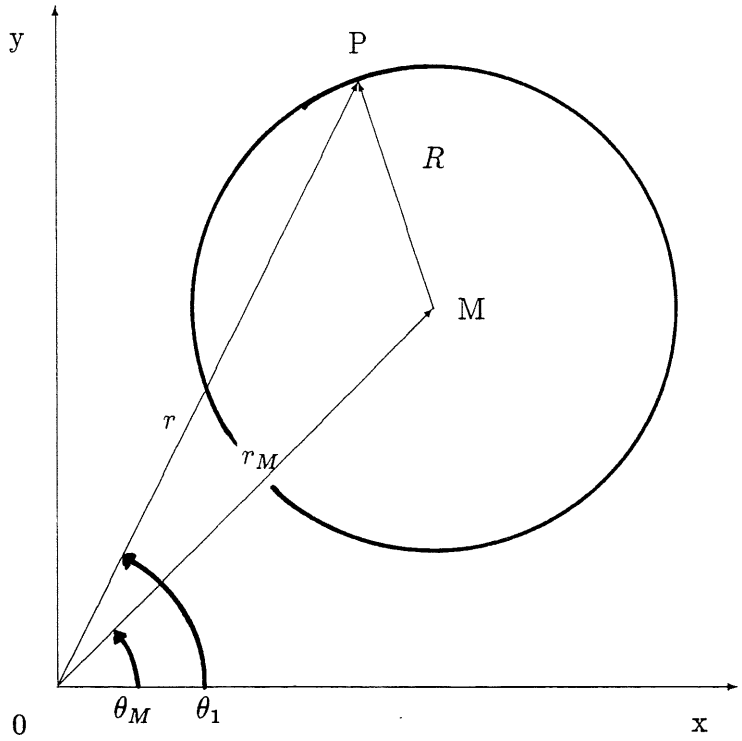


Abbildung 16: Verwendete Bezeichnungen

nachfolgenden Kammer. Der Algorithmus wird bis zur letzten Kammer im Zentrum fortgesetzt.

Die CJC ist nicht überall aktiv. So werden in manchen Kammern keine Signale ausgelöst. Als Spurkriterium kann gelten, daß mindestens in:

1. jeder dritten Kammer oder
2. daß in z.B. 19 von 24 Kammern ein Signal auftritt.

Die erste Bedingung ist günstiger für sequentiell arbeitende Prozessoren, weil bei zufälligen Signalen in der Referenzkammer der Algorithmus früher abbricht.

Wie sehen in diesem Algorithmus die auszuführenden mathematischen Operationen aus? Der Radius (R) und der Tangentenwinkel (Φ_T) werden berechnet aus der Winkeldifferenz des Punktes in der Referenzkammer mit den Polarkoordinaten (r_1, φ_1) und eines weiteren Punktes (r_2, φ_2).

Die allgemeine mathematische Gleichung eines Kreises lautet:

$$r^2 - 2 \times r \times r_M \times \cos(\theta - \theta_M) + r_M^2 - R^2 = 0. \tag{1}$$

Das Koordinatensystem wird so gewählt, daß der Punkt (0,0) auf den Kreis fällt. Als zweites soll der Punkt M auf der X-Achse liegen. Dann gilt:

$$\begin{aligned} r_M &= R\,, \\ \theta_M &= 0^\circ\,, \end{aligned} \tag{2}$$

$$\begin{aligned} \Rightarrow \quad 0 &= r^2 - 2 \times r \times R \times \cos\theta \ , \\ \Leftrightarrow \quad r &= 2 \times R \times \cos\theta \ , \\ \Leftrightarrow \quad r &= 2 \times R \times \sin\left(\tfrac{\pi}{2} - \theta\right) \ , \\ \Leftrightarrow \quad r &= 2 \times R \times \sin\vartheta \ , \\ \Leftrightarrow \quad R &= \tfrac{r}{2 \times \sin\vartheta} \ , \end{aligned} \tag{3}$$

Für die Punkte Eins (Referenzkammer) und Zwei gilt dann:

$$\begin{aligned} \Leftrightarrow \quad r_1 &= 2 \times R \times \sin\vartheta_1, \\ \Leftrightarrow \quad r_2 &= 2 \times R \times \sin\vartheta_2, \\ \Leftrightarrow \quad \tfrac{r_1}{r_2} &= \tfrac{\sin\vartheta_1}{\sin\vartheta_2} \end{aligned} \tag{4}$$

Die Winkel ϑ_1 und ϑ_2 sind nicht bekannt. Bekannt ist jedoch die Winkeldifferenz beider Winkel α:

$$\alpha = \varphi_2 - \varphi_1 = \vartheta_2 - \vartheta_1$$

so folgt daraus:

$$\frac{\sin\vartheta_1}{\sin(\vartheta_1 + \alpha)} = \frac{r_2}{r_1}.$$

Aus dem Additionstheorem [18]:

$$\sin(x + y) = \sin x \times \cos y + \cos x \times \sin y \ ,$$

folgt:

$$\sin\vartheta_1 \times \frac{r_1}{r_2} = \sin\vartheta_1 \times \cos\alpha + \cos\vartheta_1 \times \sin\alpha \tag{5}$$

$$\Leftrightarrow \quad \tfrac{r_1}{r_2} = \cos\alpha + \cot\vartheta_1 \times \sin\alpha \ . \tag{6}$$

$$\Leftrightarrow \quad \vartheta_1 = \operatorname{arc\,cot}\left(\tfrac{r_1}{r_2 \times \sin\alpha} - \cot\alpha\right)$$

Die Differenz von φ_1 und ϑ_1 ist der gesuchte Tangentenwinkel ($\phi_T = \varphi_1 - \vartheta_1$). Die Gleichung 6 setze man in Gleichung 4 ein und löse nach dem Spurradius (R) auf. Man erhält dann:

$$R = \frac{r_1}{2 \times \sin\left(\operatorname{arc\,cot}\left(\tfrac{r_1}{r_2 \times \sin(\varphi_1 - \varphi_1)} - \cot(\varphi_2 - \varphi_1)\right)\right)} \tag{7}$$

Die Gleichung 7 bestimmt den Radius aus den Koordinaten der Punkte Eins und Zwei. Um die Koordinaten eines Punktes Drei zu suchen, wird wieder der Kreisalgorithmus verwendet:

$$\begin{aligned} r_3 &= 2 \times R \times \sin(\varphi_3 - \vartheta_T) \\ \Leftrightarrow \varphi_3 &= \vartheta_T \times \arcsin\left(\tfrac{1}{2} \times \tfrac{r_3}{R}\right) \ . \end{aligned} \tag{8}$$

Weil diese mathematischen Operationen recht kompliziert sind, lassen sie sich nur durch eine Reihenentwicklung von einem Prozessor berechnen. Die auftretenden Gleichungen (6,7 und 8) können jedoch auf eine Variable reduziert werden. Die

Berechnung kann daher durch Tabellen ersetzt werden. Für jedes Verhältnis von r_1 zu r_2 ($\frac{r_1}{r_2}$) und für die Winkel α muß ein Tabellenwert für die Gleichungen (6,7 und 8) vorhanden sein. Diese Tabellen sind daher recht umfangreich.

Für den Algorithmus ist es notwendig, sowohl den Spurradius (R) als auch den Tangentenwinkel (ϕ_T) zu berechnen. Mangels Genauigkeit der Eingangswerte können diese von dem H1 Datensystem nicht weiter verarbeitet werden und sind somit kaum weiter nutzbar.

Für diesen Algorithmus werden Prozessoren benötigt, die schnell Tabellenwerte holen und Additionen ausführen. Hier ein kurzer Vergleich einiger verschiedener Typen.

Prozessortyp	**Hersteller**	**Zeit für Tabellenlesen** $\times\mu$sec.	**Zeit für Addition** $\times\mu$sec.
DSP56001	Motorola	0,2	0,1
MC68020	Motorola	≈0,55	≈0,1
T414	Inmos	0,38	0,1

Die Zeiten des MC68020 können nur grob angegeben werden, da diese stark von der Länge des betrachteten Programmes und von der Umgebung der Befehle abhängen. Die Zeiten für den DSP56001 gelten für Tabellen, die sich in prozessorexternen Speichern befinden.

Im Verlauf dieser Untersuchung wurde dieser Algorithmus in ein Assemblerprogramm für den DSP56001 und MC68020 umgesetzt. Die Zyklen des DSP56001 Programms wurden mit dem Simulator SIM56000 der 'DSP Development Software' von Motorola bestimmt. Die Zykluszahl des MC68020 Programms wurde ausgezählt. Die untersuchten Programme berechneten nur dreimal den Radius und den Tangentenwinkel. Als Speicher für die Signale wurden spezielle assoziative Speicher (CAM) simuliert (siehe hierzu Kapitel 6.1.1.). Der Tabellenumfang betrug 19,8kByte. Dieser Algorithmus wurde auch in ein Fortran77 Programm[3] (siehe Anhang VI.2.) umgesetzt, um die Laufzeit auf einem Großrechner zu erproben. Die Laufzeiten des Fortran77 Programmes wurden auf einer IBM 3084Q unter Verwendung der DESY-Biblotheksroutine TMLOG bestimmt. Das Ergebnis dieser Untersuchung ist in der nachfolgenden Tabelle dargestellt:

Prozessortyp	**Zeit pro einzelne Spur** $\times\mu$sec
DSP56001	35
MC68020	≈52
IBM 3084Q	78

[3]siehe hierzu Literatur [10]

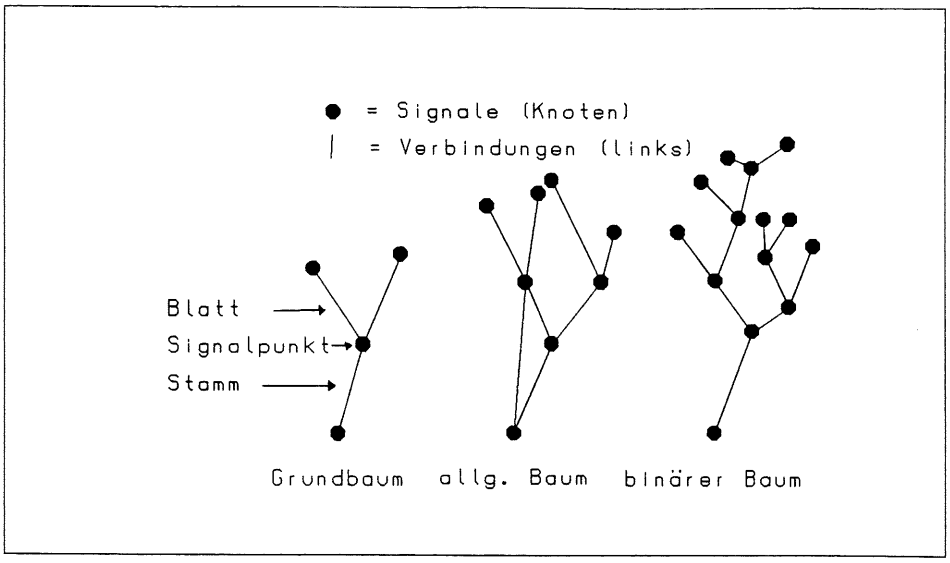


Abbildung 17: Verschiedene Bäume

4.2. Spursuche in Baumstrukturen

Der Suchalgorithmus in Baumstrukturen[4] arbeitet in drei Stufen. Als erstes baut der Algorithmus eine Liste aller Verbindungen zwischen Signalkoordinatenpaaren (englisch: *links*) auf. Als zweites ordnet der Algorithmus diese Liste, und als letztes werden die *links* zu Ketten verbunden. Die Struktur der geordneten Liste wird in der Graphentheorie "Baumstruktur" genannt. Abbildung 17 zeigt einige solche "Bäume". Jeder "link" bildet den Stamm genau eines Grundbaumes. Alle folgenden *links*, die ebenfalls die eine Signalkoordinate enthalten, bilden die Blätter. Die Grundbäume werden zu vollen Bäumen zusammengesetzt. Vollständige Spurstraßen sind als längste Kette (= größte Anzahl) von *links* im Baum erkennbar. Der Algorithmus durchläuft alle Bäume und bestimmt die darin enthaltenen Ketten. Aus der Länge der Ketten und der Länge der einzelnen *links* bestimmt der Algorithmus die besten Spurkandidaten. Da die Zahl der möglichen *links* sehr groß ist, werden an ihren Aufbau zusätzliche Bedingungen gestellt:

1. die *links* werden nach der Anzahl der Lücken (Kammern, in denen keine Signale gefunden wurden) vorgeordnet.

2. Es werden nur solche *links* berücksichtigt, deren Winkel zum Stamm kleiner ist als ein Maximalwinkel (dies dient zur Festlegung eines minimalen Impules).

3. Nicht alle *links* werden in einer Kammer sofort konstruiert, sondern es werden schrittweise nur einzelne Sektoren untersucht. Besondere Beachtung gilt daher den Spuren, die über Sektorgrenzen hinweggehen.

[4] siehe auch Literatur [11], [12] und [14].

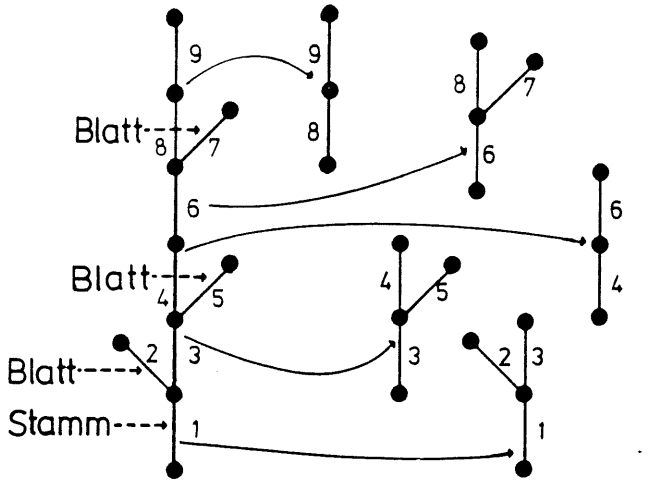


Abbildung 18: Funktion des "Kletterns"

4. Nachdem eine Spur gefunden worden ist, werden alle verwendeten *links* und deren rechte oder linke Nachbarpunkte (Spiegelpunkte) gelöscht. Sie stehen somit anderen Spuren nicht mehr zur Verfügung.

Diese *links* werden zu Grundbäumen aufgebaut und in einer Liste zusammengefaßt. Das Sortieren der Liste dieser Grundbäume geschieht durch einen vergleichenden Sortieralgorithmus. Dieser Algorithmus benutzt die Datenstruktur des binären Baumes und wird "heapsortprocedure" (Haufensortierung) genannt. Der Aufbau der Ketten geschieht durch einen Algorithmus mit dem Namen "depth first search". Dieser Algorithmus erklettert den Baum, ausgehend von einem Start-*link*, ohne daß Ketten doppelt durchlaufen werden. Dieser Vorgang wird in Abbildung 18 skizziert. Zur Bestimmung der Zeit für die Rekonstruktion einer Spur wird die Zahl der benötigten elementaren Operationen ausgezählt.

4.2.1. Das Aufbauen der Verbindungspaare

Der Algorithmus soll auch inaktive Kammern überbrücken und so müssen auch *links*, die über mehr als eine Kammer reichen, aufgebaut werden. Als Beispiel betrachten wir hierbei alle *links*, die eine, zwei oder drei Kammern überbrücken. Der Algorithmus muß unter der Annahme, daß die Spuren nicht zu stark gekrümmt sind, bei jedem Signal an 9 Koordinaten nach weiteren Signalen suchen.

Bei einer Spur mit 24 Signalen ergeben sich 23×3 $(= 69)$ *links* ' und 23×9 $(= 207)$ Suchvorgänge.

Jede Suchoperation besteht aus einem Befehl, der ein Datum holt, und einem vergleichenden Befehl. Jeder Befehl zum Listenaufbau führt eine Schreiboperation aus. Für die verschiedenen Prozessoren ergeben sich folgende Zeiten:

○ = Suchkoordinate
↑ = mögliche *link's*
● = Signalpunkt

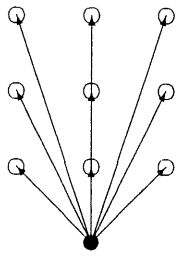


Abbildung 19: Signal mit Suchkoordinaten

Prozessor	**Zeit für die Suche** $\times\mu$sec	**Zeit für das Aufbauen** $\times\mu$sec
DSP56001	0,20	0,1
MC68020	≈0,24	≈0,16
T414	0,26	0,21

Somit ergibt sich eine Gesamtzeit pro einfacher Spur von:

Prozessor	**Zeit** (T_{Spur}) $\times\mu$sec
DSP56001	48,3
MC68020	≈60,72
T414	68,31

4.2.2. Das Sortieren der Liste der Verbindungspaare

Aus diesen *links* müssen geordnete Listen aufgebaut werden. Die verwendeten Algorithmen sehen wie folgt aus:

Unterprogramm **Ordnen(i, j)**

falls i kein Blatt des binären Baumes ist, und falls ein Nachfolger von i größer ist als i, dann mache folgendes:

falls k der Nachfolger mit dem größten Element ist,
dann tausche k mit i und
rufe **Ordnen(k, j)** *auf!*

Unterprogramm **Liste aufbauen**

Für i=n bis 1 führe **Ordnen(i,n)** *aus!*

Programm **Sortieren**

führe **Liste aufbauen** *aus!*
für i=n bis 2 mache folgendes:

tausche 1 mit i
führe **Ordne(1,i-1)** *aus!*

Das n steht für die Anzahl der Elemente der Liste.
Jeder Durchlauf besteht mindestens aus folgenden Operationen:

69×	Vergleiche in **Liste aufbauen** bzw. **Ordnen**
68×	Vergleiche in **Sortieren** bzw. **Ordnen**
68×	Tauschen in **Sortieren**

Zu beachten ist, daß eine Tauschoperation aus drei Befehlen zur Datenverschiebung und eine Vergleichsoperation aus einem vergleichenden Befehl und zwei Schieboperationen für die Operanden besteht. Folgende Zeiten ergeben sich dann bei den verschiedenen Prozessoren:

Prozessor	**Zeit für Tausch** $\times\mu$sec	**Zeit für Vergleich** $\times\mu$sec
DSP56001	0,30	0,30
MC68020	≈0,64	≈0,40
T414	0,63	0,56

Somit ergibt sich eine Gesamtzeit pro einfache Spur von:

Prozessor	**minimale Zeit** (T_{Spur}) $\times\mu$sec
DSP56001	62,5
MC68020	98,3
T414	119,6

4.2.3. Das Aufbauen der Ketten

Dieser Algorithmus sieht wie folgt aus:

Programm **Aufbau der Ketten**

markiere verwendeten Knoten als verwendet!

für jeden Nachfolger des verwendeten Knotens mache folgendes:

falls der Nachfolger noch nicht markiert ist,
führe folgendes aus:

hänge diesen Nachfolger an die Knotenliste!
rufe Programm **Aufbau der Kette** *auf!*

Im günstigsten Fall benötigt dieser Algorithmus für eine vollständige Spur dreiundzwanzig Durchläufe. Jeder dieser Durchläufe besteht aus einem Vergleich und zwei Schreibbefehlen. Es ergeben sich folgende Zeiten pro Spur:

Prozessor	**Zeit für Durchlauf** $\times\mu$sec	**Gesamtzeit** $\times\mu$sec
DSP56001	0,30	6,90
MC68020	$\approx$0,40	$\approx$9,20
T414	0,56	12,88

4.2.4. Eigenschaften der Spursuche in Baumstrukturen

Wenn man nun die Zeiten für das Aufbauen der *links*, das Sortieren der *links* und das Durchsuchen des *Baumes* für eine einfache, vollständige Spur addiert, kommt man zu folgenden Mindestzeiten pro konstruierter Spur:

Prozessor	**Gesamte Zeit** (T_{Spur}) $\times\mu$sec
DSP56001	116,6
MC68020	168,22
T414	200,79

Dieser Algorithmus ist geeignet für die entgültige Spur-Rekonstuktion eines Ereignisses. Als solcher ist dieser bei dem Experiment TASSO bei DESY eingesetzt worden[5]. Er bewies eine große Effizienz und benötigte bei 8 Kammern pro Ereignis $\approx$40ms auf der IBM 3081. Als 3. Triggerstufenprogramm ist er weniger geeignet, da er sehr viel Zeit benötigt.

4.3. Spursuche im reziproken Raum

Wie in den vorangegangenen Kapiteln beschrieben, müssen Algorithmen im realen Raum Signalpunkte auf Kreissegmenten suchen. Der Wechselwirkungspunkt liegt in erster Näherung ebenfalls auf diesen Kreissegmenten. Bei der Wahl eines Koordinatensystems mit dem Wechselwirkungspunkt als Ursprung kann die Kreisverwandtschaft zwischen Kreisen und Geraden zur Spursuche ausgenutzt werden[6]. Jeder Kreis und jede Gerade in der Ebene kann durch eine Gleichung der Form

$$\alpha(x^2 + y^2) + \beta x + \gamma y + \delta = 0, \tag{9}$$

[5] siehe hierzu Literatur [11] und [12]
[6] siehe hierzu auch Literatur [19]

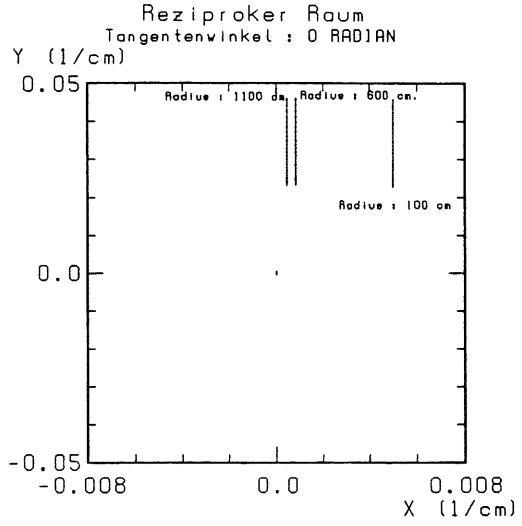


Abbildung 20: Darstellung von Spuren im reziproken Raum

dargestellt werden, wenn in ihr α, β, γ und δ passende reelle Konstanten bedeuten und

$$4\alpha\delta < \beta^2 + \gamma^2 \tag{10}$$

gilt. Die reziproken Koordinaten können durch Gleichungen der Form

$$x' := \frac{x}{x^2 + y^2} \quad , \quad y' := \frac{y}{x^2 + y^2} \tag{11}$$

definiert werden, wobei ungestrichene Variablen Koordinaten im realen Raum darstellen. Falls der Punkt (x,y) der Gleichung 9 genügt, erfüllt Punkt (x', y') nach Gleichung 11 folgende Gleichung:

$$\alpha\frac{x'^2 + y'^2}{(x'^2 + y'^2)^2} + \beta\frac{x'}{x'^2 + y'^2} + \gamma\frac{y'}{x'^2 + y'^2} + \delta = 0 \tag{12}$$

oder genügt der einfacheren Gleichung:

$$\alpha + \beta x' + \gamma y' + \delta(x'^2 + y'^2) = 0 \tag{13}$$

und liegt daher wieder auf Kreisen oder Geraden. Daraus folgt, daß jedem Kreis im realen Raum, der durch den Nullpunkt (0,0) geht ($\alpha \neq 0, \delta = 0$), eine Gerade im reziproken Raum entspricht, die nicht durch den Nullpunkt geht und zur Tangente des Kreises in (0,0) parallel ist. Der Abstand dieser Geraden zum Pol (0,0) entspricht dem reziproken Wert des Radius des verwandten Kreises. In Abbildung 20 und 21 sind nun drei simulierte Spuren im realen und im reziproken Raum dargestellt. Die Suche von Kreissegmenten im realen Raum wird transformiert in die Suche von Geraden im reziproken Raum.

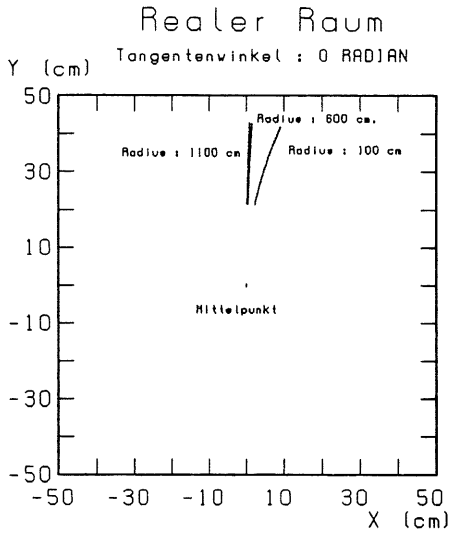


Abbildung 21: Darstellung von Spuren im realen Raum

Wie vollzieht sich die Transformation der Koordinaten der Meßwerte vom realen in den reziproken Raum? Der Winkel φ bleibt unverändert

$$\varphi' = \varphi \quad , \quad R' = \frac{1}{R} \tag{14}$$

und der Radius wird reziprok. Es treten jedoch bei der Erstellung effektiver Suchalgorithmen im reziproken Raum mehrere Probleme auf:

- Bei der Verwendung von Polarkoordinaten müssen trigonometrische Funktionen verwendet werden, denn die Geraden werden in der Form

$$r = \frac{p}{\sin(\varphi - \alpha)} \tag{15}$$

  dargestellt, mit p als Abstand der Geraden von dem Pol (0,0), α als dem Winkel zwischen der Polachse und der Geraden (Tangentenwinkel) und (r,φ) als der Koordinate eines Punktes auf der Geraden. Die Behandlung dieser trigonometrischen Funktionen erfordert der Rechenzeit wegen umfangreiche Tabellen.

- Der reziproke Wert der Radien muß errechnet werden.

- Bei der Verwendung von Parallelkoordinaten müssen entweder drei Punkte verwendet werden, um die Gerade nach der Gleichung

$$ax + by + c = 0 \tag{16}$$

  festzulegen, oder man verwendet zwei Punkte zur Charakterisierung der Geraden nach der Gleichung

$$y = mx + n, \tag{17}$$

doch muß dann noch eine Ausnahmeroutine installiert werden, welche bei *m> Maximalwert* eine Drehung des Koordinatensystems durchführt. Dies führt jedoch zu einer erheblichen Erhöhung der Rechenzeit.

Die genannten Punkte zeigen, daß Suchalgorithmen im reziproken Raum einen großen Rechenaufwand erfordern. Daher wurde kein Algorithmus gefunden, welcher in der in dieser Arbeit diskutierten Zeit von 0,5msec ein brauchbares Ergebnis lieferte.

4.4. Spursuche mit einem "Masken"- Prozessor

Dieser Algorithmus[7] ist eigentlich kein reiner Softwarealgorithmus. Er benötigt einen speziellen Prozessor. Der Algorithmus geht aus von einem großen "Schalterkasten". Dieser Schalterkasten besitzt entsprechend der Auflösung der Driftstrecken der CJC 24 Reihen a 800 senkrechte[8] und 24 Reihen a 800 waagerechte Schalter. In Bild 22 ist symbolisch ein solcher Schalterkasten dargestellt. Die Treffer werden in einen solchen Schalterkasten durch Schließen der Schalter um den Trefferpunkt eingetragen. Durch Anlegen einer Spannung an der unteren Leiste A kann an der Leiste B die Zahl der vollständigen Spuren geprüft werden. In der Abbildung 23 ist ein Ereignis in einem Maskenprozessor dargestellt.

Wie bereits in Kapitel 2.2. erwähnt, ist die CJC nicht immer überall aktiv. Spuren sollen zudem nur bei einem Mindestimpuls rekonstruiert werden. Die Wahl der Schalter, die man bei dem Eintrag eines Signales schließt, Maske genannt, ermöglicht die Berücksichtigung des minimalen Impulses und der Inaktivität. In Abbildung 24 und 25 sind Masken zur Überbrückung inaktiver Kammern und zur Ablehnung von Spuren mit zu kleinen Impulsen skizziert. Dieser Algorithmus kann nicht definitiv Kreissegmente, sondern nur Straßen, welche grobe Bedingungen erfüllen, rekonstruieren.

Der Schalterkasten wird auf dem Spezialprozessor verwirklicht. Die Schalter werden aufgebaut aus einem Und- Gatter mit mindestens sechs Eingängen und einem Flip-Flop[9], welcher den Zustand "geschlossener Schalter" oder "offenener Schalter" speichert (siehe Abbildung 26). Diese Flip-Flops werden gesetzt von einer freiprogrammierbaren, einfachen Logik. Durch diese programmierbare Logik kann der Benutzer eines solchen Prozessors die Masken seiner Anforderung entsprechend setzen. Diese Logik steuert den gesamten Schalterkasten. Sie gleicht auch Totalausfälle bestimmter Komponenten aus. Ein weiteres externes Mikroprozessorsytem kann die gefundenen Straßen, unter Verwendung eines der anderen Algorithmen, weiter qualifizieren. Ein solcher Spezialprozessor wird in dem Fortranprogramm in Anhang VI.1. simuliert. Es ergeben sich unter der Annahme eines Prozessortaktes von 25MHz folgende Zeiten:

[7] siehe hierzu auch Literatur [15] und [16]

[8] bei einem Radius von 43cm und einer Driftstrecke von 3,5mm pro Auslese ergeben sich ca. 800 mögliche Winkel

[9] logischer Baustein, welcher gesetzt und rückgesetzt werden kann, und diesen Zustand speichert

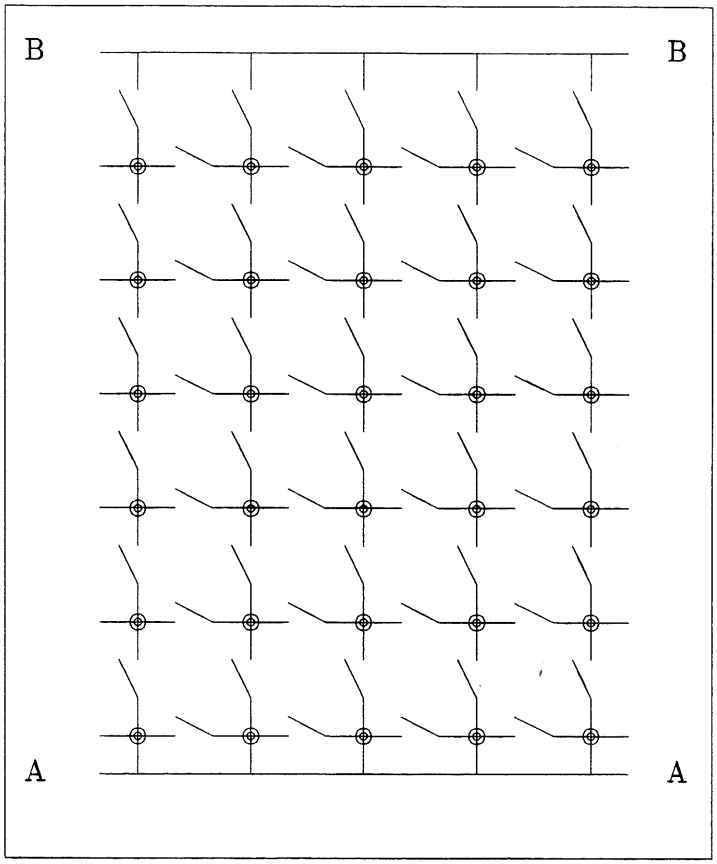


Abbildung 22: symbolische Darstellung eines Schalterkastens

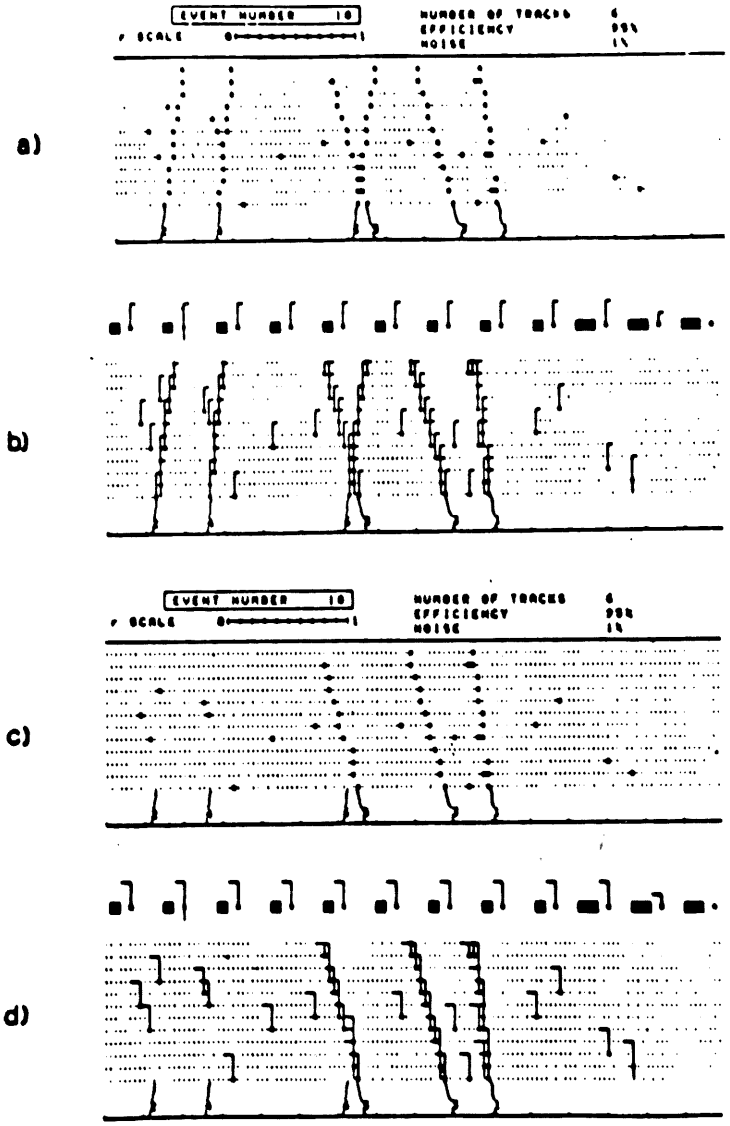


Abbildung 23: R-Φ Projektion eines Ereignisses im Maskenprozessor

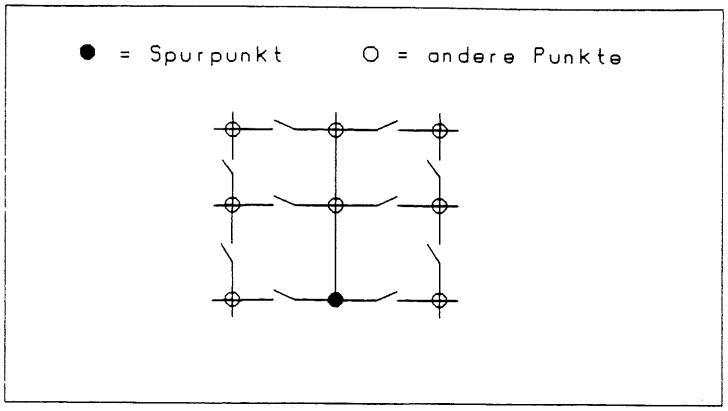


Abbildung 24: Maske zur Überbrückung inaktiver Kammern

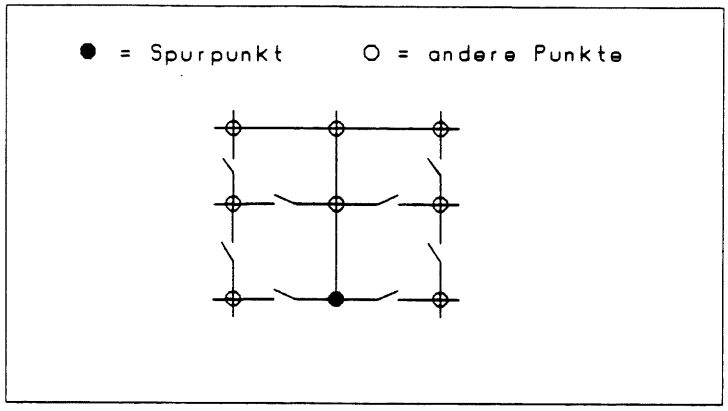


Abbildung 25: Maske für schwach gekrümmte Spuren

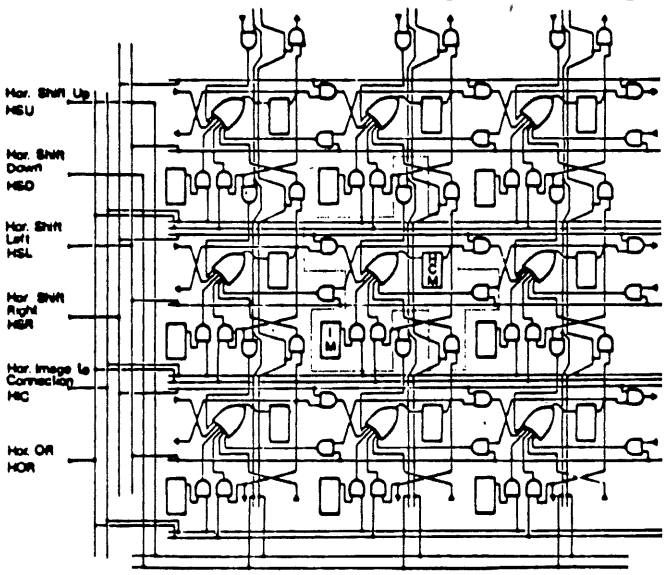


Abbildung 26: Technische Realisierung der "Schalter".

	Aufgabe	**Zeit** $\times\mu$sec
1.	Finden der Spuranzahl	1,0
2.	Trennen der Spuren	1,0
3.	Masken festlegen	4–12

Dieser Algorithmus besitzt als Nachteil die hohen Entwicklungskosten des Spezialprozessors, die schlechte Spur-Rekonstruktion und die ungewöhnliche Programmierung.

4.5. Begrenzter *Football*- Algorithmus

In Abbildung 27 sind einige mögliche Spuren aus dem Wechselwirkungspunkt dargestellt. Alle besitzen ein gemeinsames Referenzsignal in der Referenzkammer. Da die Fläche, die begrenzt wird von den Spuren mit dem minimal zulässigen Transversalimpuls, einem amerikanischen Fußball ähnelt, nenne ich sie "football". Die Zahl der unterscheidbaren Spuren ist abhängig von der Auflösung der Kammern. Bei einer Breite der Signalpunkte von 3,5mm, beträgt die Auflösung in dem *football* bei einem minimalen Transversalimpuls von $350\frac{\text{MeV}}{\text{c}}$ elf Spuren, fünf positiv, fünf negativ und eine nicht gekrümmte. Aus der Winkeldifferenz eines Signals in der Referenzkammer und eines Signals einer anderen Kammer kann durch einen Lesevorgang in einer Tabelle die Nummer der durch diese Punkte gehende Spur bestimmt werden. Jede Tabelle besteht durch die begrenzte Spuranzahlen nur aus ca. zwanzig Einträgen.

Mit den Signalen der Referenzkammer werden nacheinander folgender Ansätze gemacht:

> *Wenn ein Signal zu einer vollständigen Spur gehört, so muß in einem Bereich um diesen Punkt in der darunterliegenden Kammer auch ein Signal zu finden sein. Dieser Bereich hängt ab von dem minimal zulässigen Impuls der Teilchen. Läßt sich ein solcher Punkt finden, so kann man aus der Winkeldifferenz zum Referenzkammersignal die Spurnummer bestimmen. Aus dieser Spurnummer berechnet sich die Winkeldifferenz zum Fortsetzungspunkt in der nachfolgenden Kammer. Der Algorithmus wird bis zur letzten Kammer im Zentrum fortgesetzt.*

Zur Simulation dieses Algorithmuses wurde ein Programm verwendet, welches nur fünfmal die Spurnummer bestimmte. Als Referenzkammer wurde die vierundzwanzigste Kammer der inneren CJC verwendet. Zur Suche eines Signalpunktes wurden 1–2 Lesevorgänge in Tabellen benötigt. Zusammmen betrug die Länge der Tabellen 0,5kByte. Als Speicher für die Signaldaten wurde ein spezieller assoziativer Speicher (CAM) simuliert.

Dieser Algorithmus ist gut geeignet für Prozessoren, die schnell Additionen ganzer Zahlen ausführen können, da häufig Winkeldifferenzen verwendet werden. Die Anzahl der Lesevorgänge in Tabellen ist gering. In der folgenden Tabelle sind die

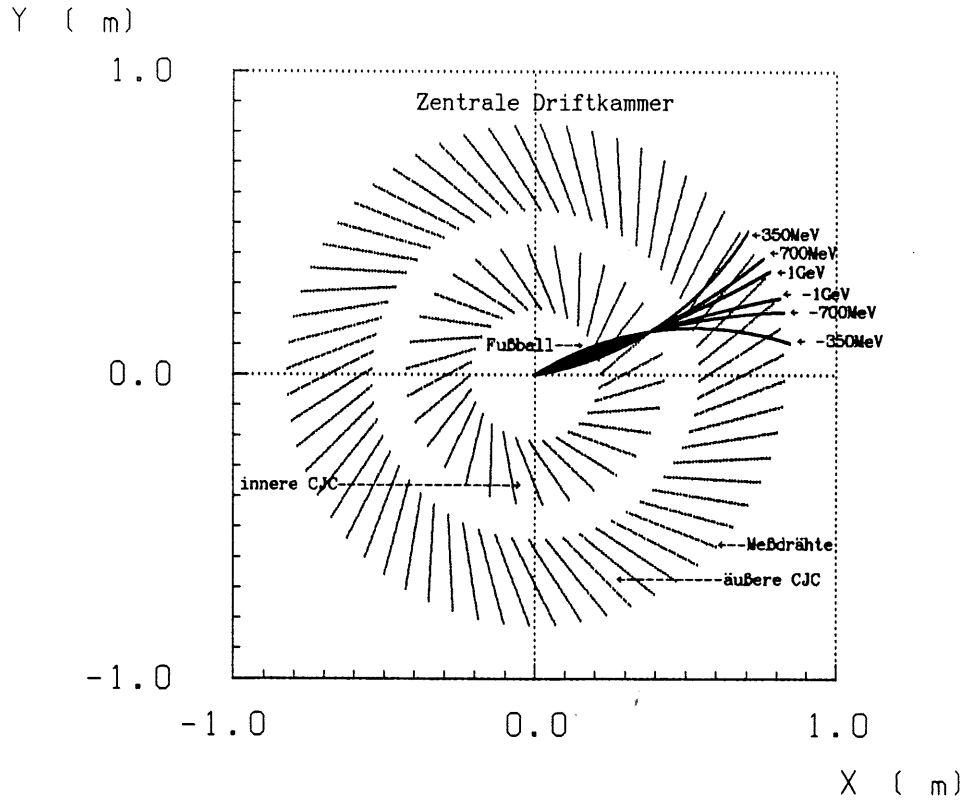


Abbildung 27: Aussehen eines begrenzten *footballs*

benötigten Zeiten pro vollständiger Spur-Rekonstruktion für zwei Prozessoren aufgeführt. Die Zeiten des DSP56001 sind durch das Simulationsprogramm SIM56000 der 'DSP Development Software' von Motorola bestimmt worden. Die Zeiten des MC68020 sind durch die Auszählung der Zyklen pro benötigten Befehls bestimmt worden. Bei dem in Anhang V.2. befindlichen Programm wurde die Routine zur "CAM"-Effizienzverbesserung in der Zeit nicht berücksichtigt, da diese bei den anderen Algorithmen auch keine Verwendung fanden.

Prozessortyp	**Zeit pro einzelner Spur** $\times\mu$sec
DSP56001	28
MC68020	$\approx$30

4.6. Zusammenfassung

Algorithmus	**Zeit pro Spur** (für DSP56001) $\times\mu$sec	**Entwicklungs-Aufwand** Mann-Jahre	**Tabellen-Länge** kByte
Kreisgleichung	35	1	19,8
Baumstruktur	48.3	1	–
Maskenprozessor	14	2	–
Football	28	1	0,5

Da ein Triggersystem der dritten Stufe bei dem Bau des H1-Detektors nicht mit hoher Priorität behandelt wird, sollte es in seinem Aufbau und seiner Entwicklung her einfach und preisgünstig sein. Diese Bedingungen werden sicher nicht durch den Maskenprozessor erfüllt. Der zweite wichtige Punkt eines solchen Triggers ist die benötigte Zeit für eine Bewertung des physikalischen Wertes eines Ereignisses. Der Suchalgorithmus in Baumstrukturen hat einen zu hohen Zeitbedarf. Zwei Algorithmen sind eine nähere Betrachtung wert. Die Entscheidung fiel auf den begrenzten *Football*–Algorithmus, da dieser durch die kleineren Tabellen weniger aufwendig ist, als der Kreisgleichungsalgorithmus.

5 Verschiedene Prozessortypen

5.1. Der Digitale Signalprozessor DSP56001 von Motorola

Der DSP56001 der Firma Motorola ist ein digitaler Signal- Prozessor (DSP). Die DSP's sind entwickelt worden zum preisgünstigen Aufbau von digitalen Filtern (z.B. schnelle Fourier Transformation 'FFT'). Sie unterscheiden sich im Allgemeinen von universellen Prozessoren durch folgende Eigenschaften:

- Zum schnelleren Datentransport besitzen DSP's intern mehrere unabhängige Busse (meist 6-7 z.B. 4 Datenbussse, und 3 Adressbusse). Jeder dieser Busse besitzt im allgemeinen eine eigene Adress- Berechnungs- Einheit (englisch: control adress unit, CAU). Diese Struktur wird auch Harvard Struktur genannt.

- Schnelle Hardware-Multiplizier- und Addiereinheiten (englisch: data arithmetik unit, DAU) liefern schon nach einem Taktzyklus ein Ergebnis.

- Zur Kommunikation mit anderen digitalen Geräten besitzen DSP's serielle und parallele Schnittstellen. Letztere können auch als 8 Bit Bus zur Kommunikation zu einer übergeordneten Recheneinheit (parallel Host MPU/DMA Interface) verwendet werden.

- Konstantenspeicher (Nur Lese Speicher : ROM) und Schreib Lese Speicher (RAM) sind schon auf dem Prozessorbaustein integriert und verringern hierdurch die externe Busbelastung.

Die Leistungssteigerung solcher Prozessoren gegenüber Standard- Prozessoren wird erreicht durch drei Konzepte. Eine Harvard Struktur ermöglicht das gleichzeitig Adressieren,Verarbeiten und Verschieben der Daten. Als zweites ermöglicht ein eingeschränkter Satz an Instruktionen (engl.: reduced instruction set, RISC) eine Verringerung der belegten Halbleiterfläche des Prozessors. Auf diese freie Fläche werden aufwendige, parallel arbeitende DAU's, Speicher und Schnittstellen aufgebaut. Diese Integration ermöglicht es, bei gleicher äußeren Anschlußzahl, die erwähnte Harvardstruktur zu verwirklichen. Als drittes werden für häufig auftretende Funktionen eigenständige Blöcke geschaffen, welche die Funktionen ausführen, ohne die Programmsteuereinheit zu belasten. So können zeitgleich verschiedene Funktionen abgearbeitet werden. Als Beispiel seien aufgeführt die Schnittstellen, das Schreiben von Daten in Register und die Arithmetikeinheiten.

5.2. Der Prozeßrechner MC68020 von Motorola

Der MC68020 von Motorola ist ein universeller Prozessor, welcher für die industrielle Prozeßsteuerung entwickelt wurde. Daher besitzt er im Gegensatz zu anderen Prozessoren dieser Leistungsklasse folgende Eigenschaften:

- Er ist mit einem externen Daten- und Adressbus versehen, welcher wechselseitig synchronisiert wird (asynchroner Bus). Somit ist eine große Anpassungsfähigkeit an verschiedene Peripherien gewährleistet. Zudem ist der externe Bus flexibel in der verwendeten Datenbreite. So kann 8 bit-, 16 bit- und 32 bitweise übertragen werden.

- Um die externe Busbelastung zu verringern, besitzt der MC68020 einen schnellen internen Speicher für häufig verwendete Instruktionen (englisch: instruction cache). So können z.B. kleine Schleifen vollständig in dem 256 Byte großen Instruktionspeicher ablaufen. Die Zugriffszeiten können sich in solchen Fällen halbieren bis dritteln.

- Der MC68020 wertet gleichzeitig bei der Ausführung einer Instruktion die drei folgenden Instruktionen (drei folgende Bytes) aus (englisch: instruction prefetch). Somit wird die Zeit zur Dekodierung der folgenden Instruktion verkürzt.

- Die arithmetischen Einheiten sind sehr universell und schnell. Die verwendete Datenbreite beträgt 32 Bit.

- Der MC68020 ist eine Weiterentwicklung des 16 Bit MC68000. Durch diese Tatsache und der großen Verbreitung dieses Prozessors existieren eine Menge Entwicklungsumgebungen und Entwicklungshilfen für den MC68020.

- Der VME Bus ist ein Bus, welcher mit geringem Aufwand mit dem MC68020 verbunden werden kann.

- Die Industrie bietet eine große Anzahl fertiger Platinen mit MC68020 Prozessoren an.

Der Einsatz dieses Prozessor wird stark begünstigt durch das häufige Vorkommen dieses in wissenschaftlichen Anwendungen. Es ist zum Beispiel geplant, einen MC68020 für die Magnetstromregelung von DESY3 zu verwenden. Durch diese Tatsache ist auch der Assembler des MC68020 weit verbreitet und auch das Angebot an höheren Programmiersprachen sehr groß. Der Einsatz dieses Prozessors bereitet daher sowohl technisch wie in der Programmierung wenig Aufwand.

5.3. Transputer von Inmos (IMS T414)

Die Transputer der Firma Inmos sind Prozessoren, welche entwickelt worden sind zur Vereinfachung von Systemen mit mehreren Prozessoren (engl.: multi processor systems). Sie unterscheiden sich von universellen Prozessoren durch folgende Eigenschaften:

- Zur Verwaltung von mehreren parallel ablaufenden Prozessen auf einem Transputer (englisch: multitask) besitzen die Transputer einen festverdrahteten Verteilmechanismus (englisch: scheduler).

- Die Kodierung der Instruktionen ist so gestaltet, so daß sie schnell ausgewertet werden können. So liegt zum Beispiel die Länge des Operatorfeldes (4Bit) und des Datenfeldes (4Bit) bei jeder Instruktion fest.

- Bedingt durch den vorherigen Punkt, besitzen Transputer einen stark eingeschränkten Instruktionssatz. Diese wenigen Instruktionen sind sehr elementar. Komplexe Instruktionen treten nicht auf. Eine solche Struktur heißt RISC (englisch: reduced instruction set).

- Um schnell mit anderen Transputer kommunizieren zu können, besitzen die Transputer außer den Adress- und Datenleitungen noch vier spezielle Schnittstellen (englisch: links). Über diese Links tauschen die Transputer mit anderen Prozessoren Daten aus. Es findet also eine Trennung zwischen lokalen und globalen Bussen statt.

- Um die Versorgung mit einem Systemtakt bei großen Transputernetzwerken zu vereinfachen, ist der externe Takt der Transputer sehr klein (5MHz).

- Schreib- Lese Speicher (RAM) sind schon auf dem Prozessorbaustein integriert und verringern hierdurch lokale Busbelastung.

- Die Transputer besitzen eigenständig arbeitende logische Blöcke, welche nicht die zentrale Prozeßsteuereinheit belasten (z.B. die Links)

Durch drei Konzepte konnte die Leistung der Transputer gegenüber Standard Prozessoren verbessert werden. Die Benutzung eines eingeschränkten Kodes beschleunigt die Befehlsausführung. Als zweites ist der Transputer ein eigenständiges Rechnersystem mit einem eigenen Daten- und Adressbus, jedoch kann der Transputer über zusätzliche Busse mit anderen Prozessoren gleichzeitig Daten austauschen. Somit können Aufgaben parallel auf verschiedenen Prozessoren abgearbeitet werden. Durch den eigenständigen lokalen Bus behindern sich die Transputer nicht gegenseitig durch Busbelegung. Als letztes wurde die Instruktionskodierung so gestaltet, daß Hochsprachen schnell in den Maschinenkode übersetzt werden können. Die Familie der Transputer umfaßt vier verschiedene Mitglieder. Diese unterscheiden sich durch die Datenbreite, dem internen Schreib- Lese Speicher und dem Datenformat. Für die untersuchte Aufgabe ist der Transputer IMS T414 ausreichend.

6 Verwendetes Programm (für den DSP56001)

6.1. Die assoziative Speicher

6.1.1. Inhaltsadressierte assoziative Speicher (CAM)

Da die Verwendung eines Mikroprozessors als Trigger eine zeitkritische Anwendung darstellt, sind einige besondere elektronische Komponenten notwendig. Als wichtigste Komponente ist der verwendetete Speicher für die Signale zu erwähnen. Die am häufigsten verwendeten Schreib- Lese- Speicher (englisch: random access memory, RAM) sind adressenorientiert. Dies bedeutet, daß der Inhalt einer Speicherzelle keine Verknüpfung mit seiner Adresse hat. Dieser Umstand ist zu vergleichen mit einer Telefonnummer, die ja auch keine Aussage enthält über den Fernsprechteilnehmer. In einem solchen Speicher ist das Suchen nach aufgetretenen Signalen sehr zeitaufwendig. Die Zeit für das Suchen kann verkürzt werden durch eine Strukturierung der Datenadressen. Es wird eine Verknüpfung zwischen Adresse und Inhalt hergestellt. Dafür kommen im wesentlichen zwei Konzepte zum Tragen, die Hashkodierung und die Tabellenkodierung. Bei der Hashkodierung werden Eigenschaften der Daten wie Quersumme, numerische Größe usw. für eine lexikalische Ordnung verwendet, so daß die Suche sich auf eine Untermenge der Daten beschränkt. Bei der Tabellensuche wird jedem möglichen Wert eine Speicherzelle reserviert. Somit ergibt sich die Adresse eines Datums aus dem Datum plus dem Tabellenanfang.

Bei der Erstellung von Programmen für einen Trigger wurde deutlich, daß bei der Suche nach aufgetretenen Signalen immer in kleinen Winkelbereichen von ca 4-8 Werten gesucht werden muß. Für den Einsatz der Tabellenkodierung bedeutet dies entsprechend 4-8 mal Adressberechnungen, Speicherlesen und Vergleich. Durch eine geeignete Wahl der Hashkodierung, könnte dies umgangen werden, jedoch hat dies eine zeitintensive Hashdekodierung zur Folge. Daher wurde in dieser Arbeit ein anderer Weg beschritten. Es wurden statt der oben erwähnten Speicher, inhaltsadressierte Speicher (englisch: content addressed memory, CAM) verwendet. Bei diesen Speichern wird zur Adressierung keine Adresse gewählt, sondern der gesuchte Inhalt. Der Speicher meldet das Vorhandensein des Datums dann an. Doch auch dies würde 4-8 Leseoperationen erfordern, wenn diese Speicher nicht noch eine zweite Eigenschaft besitzen. Die gesuchten Daten müssen nicht an allen Stellen mit den vorhandenen Daten übereinstimmen. Es können Stellen maskiert werden. So würde zum Beispiel bei der Maskierung der niederwertigsten Stelle bei der Suche nach einer 1 eine 0 ebenfalls gesucht werden. Dieses Art des Suchens wird Assoziation mit Maske genannt. Falls nun bei dem weiteren Verlauf des Programmes festgestellt wird, daß das verwendete Datum zwar innerhalb der Maske lag, jedoch nicht zu einer Spur rekonstruiert werden konnte, so kann auch ein nicht assoziativer Zugriff erfolgen. Dies bedeutet, daß das CAM nach dem nächsten Datum sucht, welches die Bedingnung erfüllt.

Die Masken besitzen jedoch einen Nachteil. In der Umgebung der Zahl 8 suchen

Mode	W	A_0	A_1	≙	I_0	I_1	D_0	D_1
Assoziieren	X	1	1	7	1/0	1/0	0	0
Asso mit Maske	1	1	0	6	1/0	X	0	D_1
Asso mit Maske	1	0	1	5	X	1/0	D_0	0
Read	1	0	0	4	X	X	D_0	D_1
Write	0	0	0	0	1/0	1/0	I_0	I_1

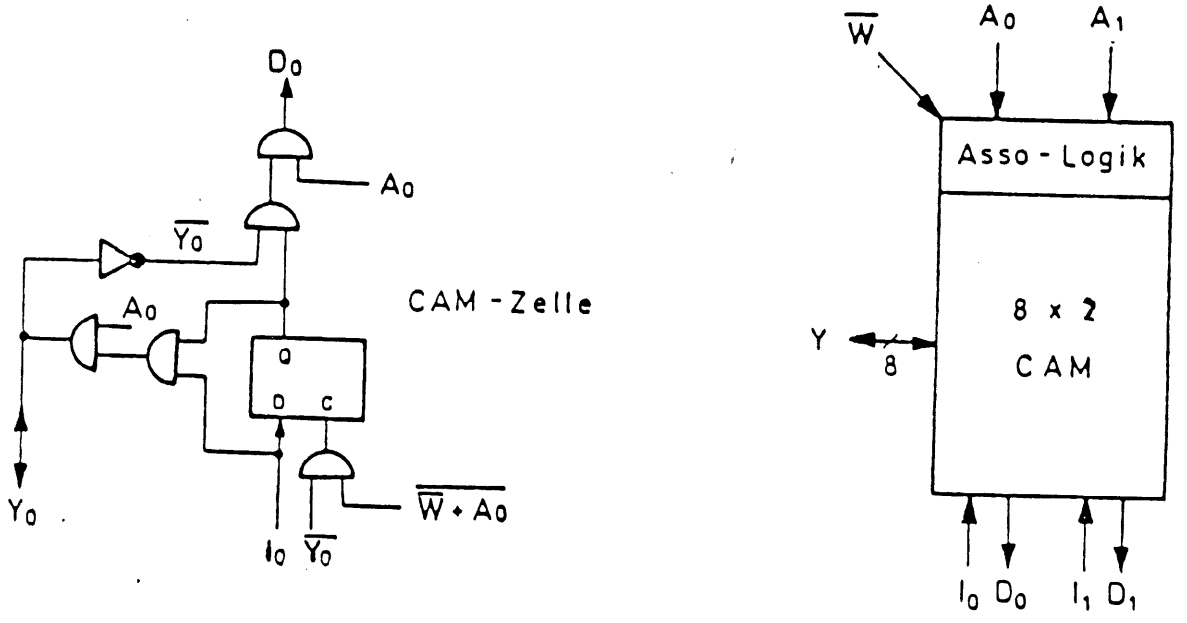


Abbildung 28: Diagramm für Signetic CAM

wir zum Beispiel Daten (z. B. 6–10). Die Zahl $8_{dezimal}$ wird binär kodiert als 1000_{dual}. Die Breite des Suchbereiches beträgt $6_{dezimal}$, also $110_{binär}$. Die verwendete Maske muß also mindestens die drei untersten binären Stellen maskieren. Dies würde aber heißen, daß die Breite des Suchbereiches nicht 6 sondern 7 Einheiten entspricht (alle Zahlen zwischen $7_{dezimal} = 111_{dual}$ und $0_{dezimal} = 000_{dual}$ können bei der Maskierung der untersten drei binären Stellen nicht unterschieden werden). Doch außer einer Verbreiterung des Suchbereiches, hat die Maskierung einen weiteren Nachteil. Wenn bei der oben erwähnten $8_{dezimal}$ drei binäre Stellen maskiert werden, reicht der Suchbereich nicht von 6–10, sondern von $8_{dezimal}$–$15_{dezimal} = 1000_{dual}$–$1111_{dual}$. Dieser Nachteil kann abgeschwächt werden, in dem nicht einmal gesucht wird, sondern zweimal. Das gesuchte Datum wird mit der halben Suchbreite subtrahiert (hier also $8 - 3 = 5$ und mit der halbe Maske gesucht. Dies entspricht dem Suchbereich von 4–7. Dann wird noch einmal gesucht, diesmal jedoch nach dem Datum $8 + 3 = 11$, was bei der selben Maske einen Suchbereich von 8–11 zur Folge hat. Insgesamt beträgt die Suchbreite also von 4–11, was dem angestrebtem Bereich von 6–10 näher kommt als die oben erwähnten 8-15. In dem vorgestellten Programm wird eine Maskierung der untersten dualen Stelle verwendet. So kann durch zwei Suchen nach vier möglichen Daten gesucht werden.

6.1.2. Die Simulation der CAMS

Da es innerhalb des Simulationsprogrammes des DSP56001 (SIM56000) keine Möglichkeit gab, als externen Speicher ein CAM zu deklarieren, mußte dieses in einer Unterroutine simuliert werden. Der Zeitbedarf dieser Unterroutine durfte jedoch nicht in den Zeitbedarf des simulierten Triggerprogrammes einfließen. Ein interner Zähler des Simulationsprogrammes wurde so eingerichtet, daß dieser die Zahl der ausgeführten Instruktionen der CAM- Simulation zählte. Diese Zähler wurde mit der mittleren Verarbeitungsdauer einer Instruktion multipliziert (ca. 3,1 Zyklen) und von dem Gesamtzeitbedarf abgezogen.

Die CAM- Simulationsroutine belegte nahezu den gesamten Y- Speicher des DSP56001. Der Y-Speicher wurde eingeteilt in 23 Bänke von je 800×24Bit Worten. Die Zahl 800 entspricht dabei dem maximal auftretenden Winkelwert (Auflösung der 24. Lage der Meßdrähte bei 96MHz. Auslesefrequenz). Diese Bänke wurden den Lagen 1–23 der Meßdrähte zugeordnet. Jedes auftretende Signal wird nun in den Bänke eingetragen. Alle übrigen Stellen erhalten eine -8388607 ($=800000_{hexadezimal}$). Bei dieser Zahl sind nämlich alle niederwertigen Bits Null, nur das 24 Bit, das als Matchsignal (ein Signal der CAMS, welches das Finden eines passenden Datums mitteilt) der CAMS dient, hat eine 1. Dies bereitet keine Datenverluste, da 800 Winkelwerte mit 10 Bits kodiert werden können, das 24. Datenbit also unbenutzt wäre.

Da die 23 Bänke á 800 Werte nur einen Speicherraum von ca 18kByte belegen, der Y- Speicher jedoch ein Adressraum von 64kByte besitzt, benutzt das Programm die oberste Adressleitung, um die CAMS, bzw. die CAM- Routine auf die gewünschte

Zugriffsart (Adressierungsart) umzuschalten (Assoziation mit Maske, Suchen ohne Assoziation) Es müssen alle auftretende Signale also doppelt eingetragen werden, in die Bank für Assoziation, und die Bank für Suchen ohne Assoziation. Die CAM-Simulationsroutine bekommt beim Aufruf nun die Anfangsadresse der Bank, in dem ein Wert gesucht wird, und den gesuchten Wert. Aus der Adresse der Bank erkennt die Routine an der obersten Adressleitung, ob eine Assoziation erwünscht ist oder nicht und verzweigt entsprechend. Falls eine Assoziation erwünscht wurde, wird der gesuchte Wert mit der Maske verundet. In einer Schleife wird nach diesem Wert in der entsprechenden Bank gesucht. Diese Schleife bricht nach der Abarbeitung der durch die Maske festgelegten Suchbreite ab und liefert bei einer Fehlsuche die oben erwähnte -8388607 ($=800000_{hexadezimal}$) als Ergebnis. Findet die Routine einen passenden Wert, so wird in einer Variablen der Zählerstand der Schleife zwischengespeichert, um bei einer Suche ohne Assoziation an dieser Stelle fortzufahren. Als Ergebnis liefert die Routine dann den gefundenen Wert, wobei das 24. Bit nun 0 ist. Das aufrufende Programm muß nun nur prüfen, ob die von der CAM- Routine gelieferte Wert negativ ist oder nicht, und erkennt daran sofort den Erfolg oder Mißerfolg einer Suche.

6.2. Die Tabellen ("lookuptables")

Bei dem gewählten 'begrenzten *Football*-Algorithmus' müssen zwei trigonometrische Gleichungen gelöst werden, um den Radius der rekonstruierten Spur und die Winkeldifferenz zwischen dem Signal in der 24. Lage und der gesuchten Lage zu bestimmen. Da die auftretenden Winkel diskreter Natur sind und die Zahl der durch die Auflösung unterscheidbaren Radien sehr klein ist, werden die Gleichungen auf eine Variable reduziert und in Tabellen im Speicher des Rechners abgelegt.

$$R = \frac{r_1}{\sin\left(\operatorname{arc\,cot}\left(\frac{r_1}{r_2 \times \sin\alpha} - \cot\alpha\right)\right)} \tag{18}$$

In dieser Gleichung sind r_1 und r_2 die Radien der Meßdrahtlagen, deren Winkeldifferenz α zur Berechnung des Radiuses verwendet wird. r_1 und r_2 sind feste Werte für gegebene Kammern. In dem verwendeten Programm ist r_1 stets der Radius der 24. Lage der Meßdrähte. Da der Radius 5 mal berechnet wird (in den Lagen 21, 20, 15, 10, 6), existieren entsprechende 5 Tabellen. Die Länge jeder einzelnen Tabelle beträgt maximal z.B. 17 Werte für einen minimalen Impuls der zu rekonstruierenden Spuren von $450\frac{\mathrm{MeV}}{\mathrm{c}}$ (Es können maximal nur 8 positiv, 8 negativ und eine nicht gekrümmte Spur aufgelöst werden). Diese Tabellen enthalten jedoch nicht wirklich die Radien, sondern es werden die Spuren durchnummeriert, die nicht gekrümmte erhält die Null, die anderen Nummern von 1–8 bzw. (-1) - (-8).

Eine zweiten Tabelle enthält für jede Lage Meßdrähte die durch die Spurnummer bedingten Winkeldifferenzen zu dem Signal in der 24. Lage. Jede dieser Tabelle hat also einen maximalen Umfang von 17 Werten.

Das Programm benutzt noch eine dritte Art von Tabellen. Diese werden benötigt, um in den Schleifen, in denen Signale gesucht werden, jedoch keine Spurnummern berechnet werden, die Adressen der zugehörigen CAMs und der Zwischenspeicher für die gefundenen Signale zu bestimmen. Diese Tabellen und die Schleifen wären zwar zu vermeiden, jedoch würde das Programm dann erheblich länger und unübersichtlicher.

Die Länge aller Tabellen betrug in der Simulation 507 Bytes. Durch diese Tabellen, speziell der zur Bestimmung der Spurnummern, kann der Benutzer eines solchen Systemes Einfluß auf den kleinsten Impuls rekonstruierter Spuren ausüben (englisch: momentum cut).

Diese Tabellen wurden in der Simulation von einem Pascal- Programm (Name: LOOKUP4A.COM) auf einem IBM-AT erzeugt. Ein späteres System könnte dieses Programm benutzen, um den Inhalt der Nur-Lese- Speicher (englisch: PROM, programmable read only memory) zu erzeugen.

6.3. Das Assemblerprogramm

Das Programm muß folgende Eigenschaften besitzen, um einen wirkungsvollen Trigger zu ergeben:

- Bei der Verwendung der CAM'S als Speicher muß die in Kapitel 6.1.1. aufgeführte Methode verwendet werden.
- Es muß auf die Überschreitung der Null°–360° Grenze geachtet werden.
- Es muß auf die unterschiedliche Winkelauflösung der einzelnen Lagen geachtet werden.
- Das Programm sollte so gestaltet sein, das es zerlegt werden kann in Unterprozesse, die eventuell von verschiedenen Rechnersystemen parallel ausgeführt werden könnten.
- Es sollte auf verschiedenen Prozessoren installierbar sein, um einen Zeitvergleich ausführen zu können.
- Es sollten die besonderen Stärken des verwendeten Prozessors unterstützen.
- Es sollten nicht zu umfangreiche Tabellen Verwendung finden, da deren technische Realisierung und das Verändern des Inhaltes der Tabellen einen größeren Aufwand erfordert.
- Es sollten die gemessenen Signale nicht mehrfach zur Rekonstruktion einer Spur heran gezogen werden (verwendete Signale löschen).
- Es sollten die Spiegelpunkte jedes Signals nicht verwendet werden.
- Es sollte die Information der Trigger der Stufe 1 und 2 Verwendung finden.

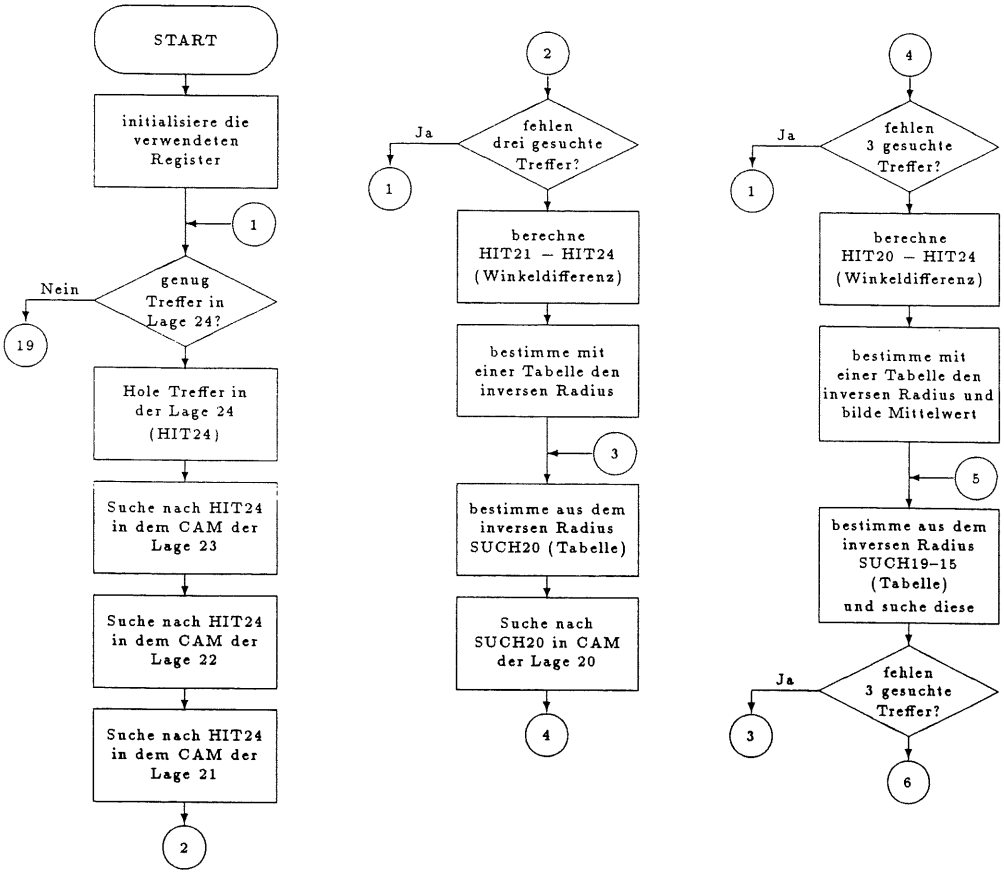


Abbildung 29: Blockflußdiagramm des verwendeten Programmes (Teil 1)

- Nachfolgenden Systemen sollen die rekonstruierten Spuren zur Verfügung gestellt werden.
- Es sollte die Möglichkeit bestehen, eine untere Grenze des zuläßigen Impulses rekonstruierter Spuren zu setzen.
- Es sollte die Möglichkeit bestehen, eine obere Grenze des dichtesten Abstandes einer rekonstruierten Spur vom Wechselwirkungs- Punkt festzulegen.
- Das Programm sollte so modular wie möglich sein, jedoch aus zeitlichen Gründen Unterroutinen vermeiden.
- Das Programm sollte nach einer frei wählbaren Zahl an rekonstruierte Spuren (Multiplizität) eine Triggerentscheidung liefern.
- Die Rate der getriggerten Ereignisse bei simulierten Monte- Carlo- Daten sollte 100Hz nicht überschreiten.
- Die Laufzeit des Programmes sollte nicht über 0,5msec. liegen.

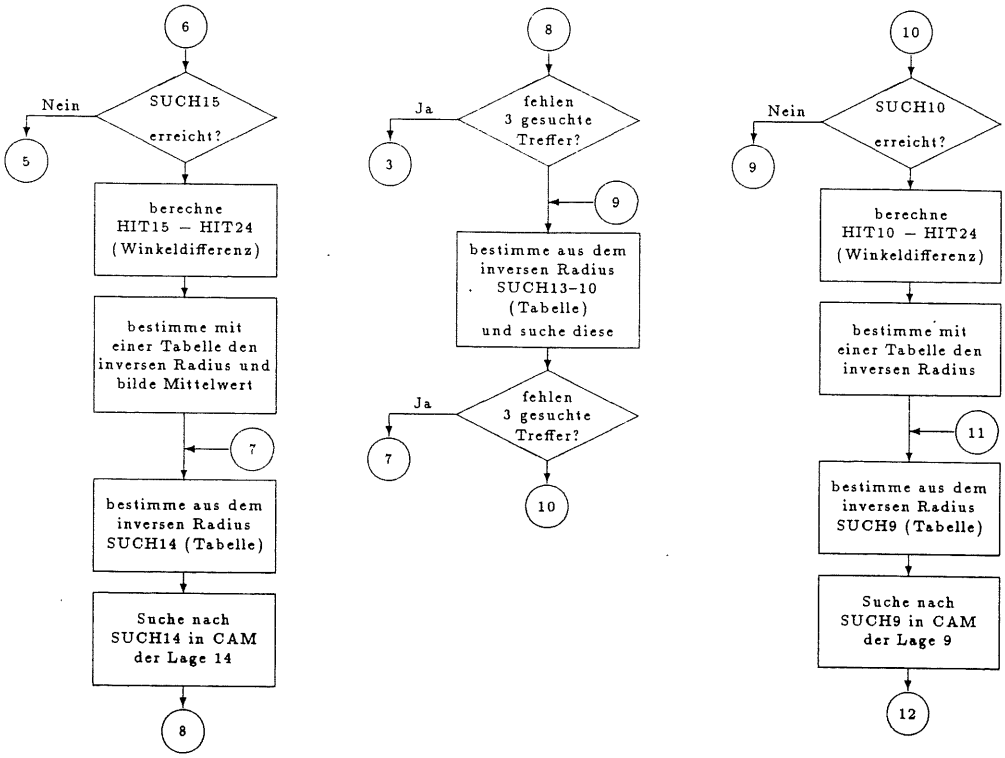


Abbildung 30: Blockflußdiagramm des verwendeten Programmes (Teil 2)

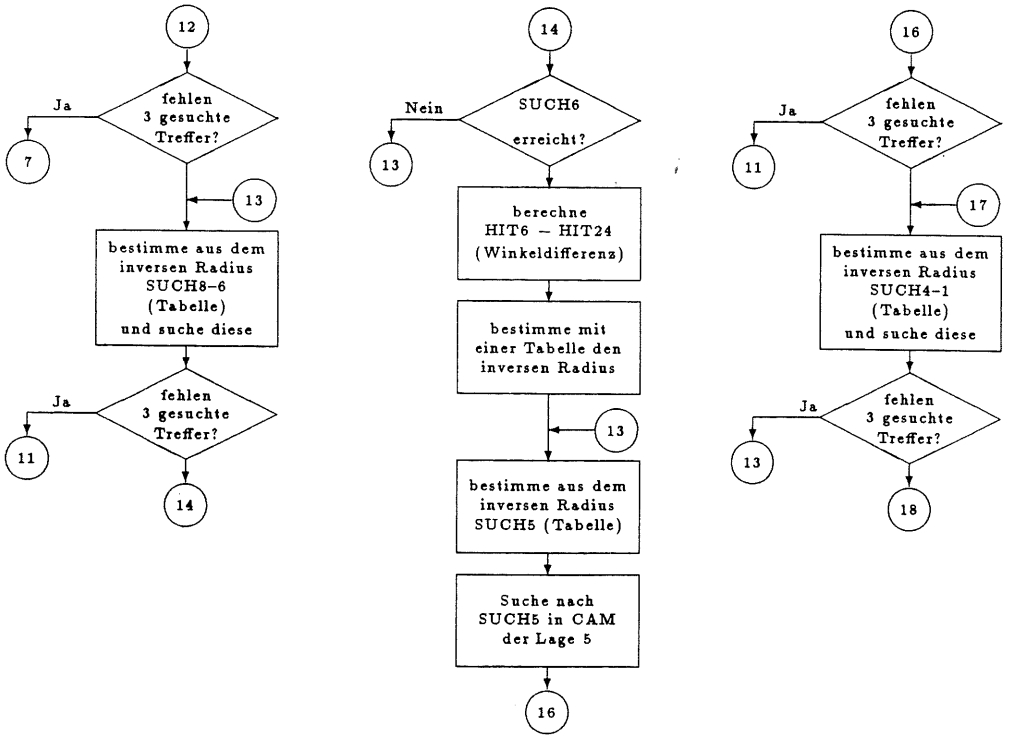


Abbildung 31: Blockflußdiagramm des verwendeten Programmes (Teil 3)

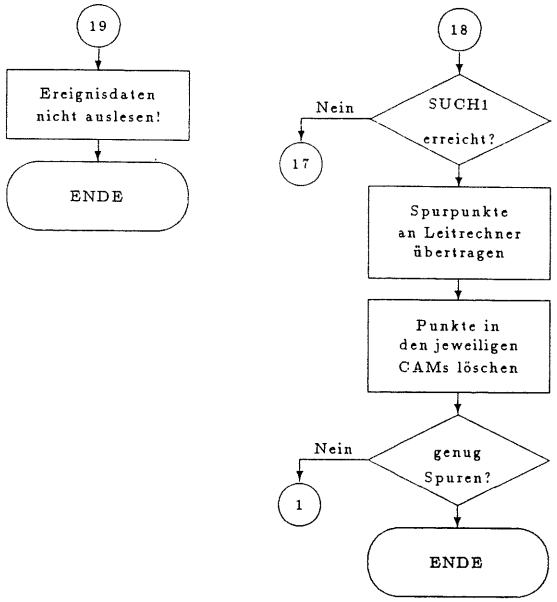


Abbildung 32: Blockflußdiagramm des verwendeten Programmes (Teil 4)

Der vorgestellte Algorithmus erfüllt die Aufgaben, bis auf die unterschiedliche Auflösung der Lagen, da diese zum Zeitpunkt dieser Arbeit noch nicht feststanden. Diese können jedoch in das Programm integriert werden.

Dieser vorgestellte Algortihmus geht davon aus, daß alle aufgetretenen Signale der inneren zentralen Driftkammer mit ihren Φ- Winkeln in den CAMs der entsprechenden Lage eingetragen sind. Die Einheit von Φ ist entsprechend der Auflösung der Lage gewählt. In der Simulation wurde durchgehend eine Auflösung von $\frac{1}{800}$ Vollkreis verwendet. Da bei der CJC Signale zweideutig sind (Driftrichtung ist nicht bekannt), werden alle Signale mit ihren Spiegelwerten paarweise eingetragen. Die Tiefe der CAMS beträgt 40 Einträge in jeder Drahtlage.

Alle Signale der 24. Meßdrahtlage werden nicht in CAMs, sondern in einem Standard Schreib- Lese- Speicher eingetragen. Zusätzlich wird in diesem Schreib-Lese- Speicher ein Zähler der Gesamtzahl an Signalen in der 24. Lage erwartet. Diese 24. Kammer ist die Referenzkammer.

Der Algorithmus arbeitet nun alle Signale in der 24. Kammer ab. Falls die 24. Kammer inaktiv oder wie Untersuchungen zeigen auf Grund von Randproblemen nicht sehr konstant in der Feldstärke ist, so kann ohne große Änderung des Programmes die 23. bzw. 22. Lage als Referenzlage dienen.

Mit Hilfe einer Maske der niederwertigsten Stelle wird nach dem Referenzsignal der 24.Lage (HIT24) in der 23–21 Lage gesucht. Alle Spuren mit Impulsen von $p > 400\frac{GeV}{c}$ liegen innerhalb der gewählten Maske. Eine Berechnung des Spurradiuses wäre auf Grund der niedrigen Auflösung des *footballs* in diesen Lagen noch nicht sinnvoll. Falls in einer Lage ein gesuchtes Signal nicht gefunden wird, so wird ein Zähler dekrementiert. Falls ein gesuchtes Signal gefunden wird, so wird der Zähler neu gesetzt (maximale Fehlstellen).

Ist in der 21. Meßdrahtlage ein Signal innerhalb der Maske gefunden worden, so wird die Winkeldifferenz zwischen diesem und dem Referenzsignal berechnet. In einer Tabelle wird aus dieser Winkeldifferenz die Spurnummer bestimmt (siehe hierzu auch Kapitel 6.2.). Aus der Spurnummer wird mit Hilfe einer zweiten Tabelle die Winkeldifferenz zwischen dem Referenzsignal und einem passenden Signal in der 20. Meßdrahtlage festgelegt. Nach diesem erwarteten Signal wird in dem CAM dieser Lage gesucht. Falls nun kein solches Signal gefunden wird, so wird der oben erwähnte Zähler dekrementiert, jedoch zusätzlich geprüft, ob dieser den Wert Null besitzt. Wenn ja, so wird das nächste Referenzsignal abgearbeitet.

Empirische Untersuchungen haben gezeigt, daß es nicht notwendig ist, in jeder Lage die Spurnummer neu zu berechnen. Daher wird die in Lage 20 bestimmte Spurnummer nun in einer Schleife benutzt, um in den Meßdrahtlagen 19–15 nach Signalen zu suchen. Falls kein passendes Signal gefunden werden kann, so wird der entsprechende Zähler dekrementiert. Falls dieser Zähler Null ist, springt das Programm zurück in die Lage, in der das letzte mal eine Spurnummer berechnet worden ist, hier also in Lage 20. Dort setzt das Programm die Suche nach einem weiteren Signal innerhalb der Maske fort. Falls kein weiteres Signal gefunden wird, springt das Signal weiter in die nächste Lage, in der eine Spurnummer berechnet wurde, hier also in die Referenzkammer.

Falls jedoch der Zähler nicht Null wurde, wird wie in der Meßdrahtlage 15 erneut eine Spurnummer berechnet, und aus der alten und der neuen Spurnummer ein Mittelwert gebildet. Dieser Mittelwert wird wiederum in einer Schleife verwendet, um in den Lagen 14–10 nach passenden Signalen zu suchen.

Nach diesem Schema wird bis zum Erreichen der 1 Meßdrahtlage verfahren. Die Signale der 10. und 6. Meßdrahtlage werden verwendet zur Bestimmung einer neuen Spurnummer.

Falls der Zähler bei vergeblichem Suchen niemals Null wird, so erreicht das Programm die 1. Meßdrahtlage. Das Rechnersystem gibt dann über den parallen Bus die Spurwerte (es könnten jedoch auch die Werte in einen Speicher mit doppeltem Zugriff, englisch: dual ported RAM, geschrieben werden) an einen übergeordneten Rechner weiter. Nach jeder erfolgreichen Spur-Rekonstruktion wird ein Zähler dekrementiert. Wird dieser Zähler Null, so fällt dieses Programm die Triggerentscheidung *"Die Ereignisdaten sollen weiter ausgelesen werden"*. Falls jedoch das letzte Signal in der Referenzkammer abgearbeitet wurde oder nicht mehr genug Signale vorhanden sind, um den Spurzähler auf Null zu setzen, so fällt das Programm die negative Triggerentscheidung *"Daten nicht weiter auslesen"*. Alle verwendeten Punkte einer rekonstruierten Spur werden in den CAMs gelöscht, in dem ein Winkel von 360°, welcher nie vorkommt und der auch in keiner Maske enthalten ist, eingetragen wird. Die Spiegelwerte werden auf diese Weise ebenfalls markiert. Dies setzt voraus, daß die Spiegelwertpaare hintereinander in dem Speicher eingetragen werden.

Als letztes muß noch erwähnt werden, daß bei der Suche nach Signalen das Programm stets überprüft, ob die 0°–360° Grenze überschritten wurde, und entsprechend zum gesuchten Winkel 360° addiert oder subtrahiert. Um bei Rücksprüngen

im Programm diese Information nicht erneut zu bestimmen, wird diese Bitweise in einer Kontrollvariablen zwischengespeichert.

Das verwendete Programm hatte eine Länge von 270 Worten (24Bit). Es diente zur Simulation des vorgestellten Triggerssystemes. Das vollständige Flußdiagramm ist in Anhang V.1. und das Assemblerprogramm in Anhang V.2. aufgeführt.

7 Die Simulations des Triggersystemes

7.1. Die verwendeten Monte–Carlo Daten

Die Monte–Carlo Daten, mit denen die Funktionsfähigkeit des hier untersuchten Triggerkonzeptes ausgetestet wurden, stammen von Herrn H.-J. Behrend (DESY-Hamburg, F36). H.-J. Behrend benutzte zur Erzeugung der Simulationsdaten für UBG's und SBG's das Programmpaket GEANT. In diesem Programmpaket wurden die spezifischen Geometrien und Komponenten des H1- Detektors und des Strahlrohres in der Umgebung des Detektors installiert. So wurden z.B. die Blenden für die Synchrotron–Strahlung, die Monitor–Platten und die Trägerkonstruktion der Zentralen– und Vorwärts–Driftkammern berücksichtigt. Da die UBG's, welche in den Detektor gestreut werden, im Strahlrohr weit oberhalb des Wechselwirkungspunktes entstehen, wurde auch das Strahlrohr bis 80m vor dem Wechselwirkungspunkt mit allen magnetischen Komponenten in das Modell integriert. Auch die Struktur der Eisenjoche der ersten 7 Quadrupole fand Berücksichtigung. Alle primären und sekundären Teilchen wurden beim Durchgang durch die Materie simuliert, bis sie einen Impuls kleiner als $50\frac{MeV}{c}$ hatten. Es zeigte sich in verschiedenen Versuchen, daß ein Senken dieser Schwelle die Rechenzeit ungemein erhöhte, ohne daß am Ergebnis sich eine deutliche Veränderung zeigte.

H.-J. Behrend hatte nicht die Abschirmung an den Stirnseiten des Detektors und um das Strahlrohr in GEANT berücksichtigt, da Versuche zeigten, daß die weitaus größte Zahl aller Untergrundereignisse durch die Öffnung des Strahlrohres in den Detektor gelangten.

Für die Erzeugung der Monte–Carlo Daten wurden zwei verschiedene Generatoren verwendet:

- Ein Programm von F.Eichler (beschrieben in [3], Kapitel 7), welches auf Messungen der Impulsspektren von Teilchenschauer bei Experimenten am Beschleuniger ISR (p–p Ereignisse) basiert.
- FRITIOF (Version 1.6), ein Generator für hadronische Kernteilchen mit kleinen transversalen Impulsen, welcher nach dem LUND-Modell arbeitet.

Die beiden Generatoren zeigen in den generierten Teilchen einige Unterschiede auf (siehe Tabelle 4). Daher wurden beide Generatoren verwendet. Es wurden

ca. 2000 Untergrundereignisse mit dem F.Eichler Progamm und ca. 1000 Untergrundereignisse mit dem FRITIOF (Version1.6, pp) Generator erzeugt. Ein Zufallsgenerator wählte einen Ereignisort aus, welcher bis zu 80m vor und 2,5m hinter dem Wechselwirkungspunkt lag. In [3] und [4] wird eine Ereignisrate für Strahl–Restgasatome von 3×10^4 Ereignisse pro 100m bei einem Vakuum von 10^{-10}mPa abgeschätzt. Diese Rate hängt empfindlich vom Druck ab. Bei der Inbetriebnahme des Speicherringes HERA ist der Druck wahrscheinlich zehnmal größer am Wechselwirkungspunkt als der oben genannte Wert. Daher gehe ich in dieser Arbeit von einer Untergrundsrate von 3×10^5Hz aus. Wenn von 3000 Untergrundereignissen eines nicht unterdrückt werden kann, so entspricht dies einer verbleibenden Untergrundrate von 100Hz ($\frac{3\times10^5\text{Hz}}{3000} = 100\text{Hz}$).

Zum Vergleich wurden auch einige Klassen physikalisch relevanter Ereignisse mit GEANT simuliert (111 Ereignisse), um zu untersuchen, ob Triggerbedingungen gefunden werden können, die den Untergrund genügend stark herabdrücken ohne viele relevante Ereignisse zu verlieren. Aus folgenden Ereignisklassen wurden physikalisch relevante Ereignisse ausgewählt:

- C.C. Ereignisse mit einem Impulsübertrag $Q^2 > 100\frac{GeV^2}{c^2}$, einer Jetenergie von $E_0 < 30$GeV und einem Jetwinkel von $\Theta_{jet} < 0,15$rad.
- N.C. Ereignisse mit einem Impulsübertrag $Q^2 > 1000\frac{GeV^2}{c^2}$ und einem Jetwinkel von $\Theta_{jet} > 0,52$rad oder $0,26\text{rad} > \Theta_{jet} > 0,16$rad
- N.C. Ereignisse mit einem Impulsübertrag $Q^2 > 1000\frac{GeV^2}{c^2}$.
- $t\bar{t}$ Ereignisse mit einem Impulsübertrag $Q^2 > 1000\frac{GeV^2}{c^2}$ und einer Masse des t–Quarks von $40\frac{GeV}{c^2}$.
- C.C. Ereignisse mit einem Impulsübertrag $300\frac{GeV^2}{c^2} > Q^2 > 100\frac{GeV^2}{c^2}$.
- C.C. Ereignisse mit einem Impulsübertrag $300\frac{GeV^2}{c^2} > Q^2 > 100\frac{GeV^2}{c^2}$ und einem x-Faktor von $0,003 < x < 0,02$.

Zusätzlich wurden noch einige 1000 geometrische Spuren mit vorgegebener Krümmung erzeugt. Diese dienten dazu, Fehler des Algorithmuses in bestimmten Winkelbereichen zu erkennen und zu beseitigen. Jeweils 10 verschiedene Spuren wurden zu einem Ereignis zusammengefaßt. Die gewählten Spurkrümmungen entsprachen Transversalimpulsen von $750–10000\frac{GeV}{c}$.

Die Koordinaten der Teilchen, welche GEANT generierte, wurden in eine Datei geschrieben. Diese Datei wurde von zwei Trigger- Simulationsprogrammen eingelesen und ausgewertet. Falls diese Programme eine positive Entscheidung fällten, wurden die Ereignisse für die weitere Simulation verwendet. Folgende Triggersimulationsprogramme wurden verwendet:

- Z- Vertex- Triggersimulation von Herrn R. Eichler, welcher die Signale der zentralen Z–Kammern zur Triggerung verwendet.

Parameter (Mittelwerte)	"Eichler"	FRITIOF Version 1.6, p–p	FRITIOF Version 1.6, p–Sauerstoff	Einheiten
Multipliziät	14,5	10,5	20,5	Teilchen
Multipliziät geladener Teilchen	7,7	5,0	19,3	Teilchen
Multipliziät von γ Teilchen	6,8	5,0	29,6	Teilchen
Impuls einzelner Teilchen	0,8	1,4	1,3	$\frac{GeV}{c}$
Θ Winkel einzelner Teilchen	0,82	0,31	0,33	Radian
maximale Energie des Ereignisses im zentralen Bereich	15	7	14	GeV
maximale Energie des Ereignisses im Vorwärts–Bereich	100	260	260	GeV

Tabelle 4: Eigenschaften der Generatoren

- Vorwärts 'ray trigger' Simulation von Herrn J.C.Bizot, welche nicht die zentrale Kammer berücksichtigte. Als Grundlage dienen hierbei die Daten der Vorwärts– Driftkammern (MWPC).

7.2. Physikalische Eigenschaften der Monte–Carlo Daten

In der zentralen Driftkammer können drei physikalische Parameter von Triggern bestimmt werden:

- Der transversale Impuls (P_T) geladener Teilchen (Ladungsabhängig).
- Die Anzahl der geladenen Teilchen (Multiplizität).
- Der kleinste Abstand der Spuren geladener Teilchen zum Wechselwirkungspunkt.

Die Monte–Carlo Daten wurden nun auf diese drei Eigenschaften überprüft, um geeignete Kriterien für einen Trigger zu finden.

In der Abbildung 33 ist das Ergebnis der Untersuchung des kleinsten Abstandes vom Wechselwirkungspunktes ($d_{Vertex\ Abstand}$) (englisch: distance of closest approach) dargestellt. Es wurden die Häufigkeit der Spuren in Prozent aufgetragen, bei denen $d_{Vertex\ Abstand}$ < Maximalwert ist. Für die relevanten physikalischen Ereignisse (englisch: real physics events) und die Untergrundereignisse (englisch: beam-gas events) wurden verschiedene Schraffuren gewählt. Bei der Betrachtung des dichtesten Abstand der Spuren geladener Teilchen vom Wechselwirkungspunkt fiel auf, daß die Verwendung dieses Parameters für einen Trigger nur sinnvoll ist, wenn der maximale Wert dieses Parameters für die Rekonstruktion von Spuren möglichst klein gewählt wird. Nur dann kann der Untergrund gegenüber dem relevanten pysikalischen Ereignissen um ca 37% (0cm < $d_{Vertex\ Abstand}$ < 0,5cm) stärker reduziert werden (siehe hierzu Abbildung 33). Bei größeren Werten dieses Parameters sinkt das Verhältnis der Reduktion auf z.B. 28% (0cm < $d_{Vertex\ Abstand}$ < 4cm). Bei der Betrachtung der Abbildung 33 fällt des weiteren auf, daß die Kurve der relevanten physikalischen Ereignisse bei 0mm erst stark ansteigt, da viele geladenen Teilchen in der direkten Umgebung des Wechselwirkungspunktes entstehen. Die Seigung der Kurve fällt dann jedoch bei ca. 1mm stark ab und nähert sich erst bei 15mm der 100%. Topologiebetrachtungen von Ereignissen mit solchen Spuren zeigten, daß diese Spuren von Paarbildungen in der Wand des Strahlrohres stammen.

In Bild 34 ist das Ergebnis der Untersuchung der Multiplizität geladener Teilchen (N_{Spuren}) der Untergrund- und physikalisch relevanten Ereignissen dargestellt. Die Häufigkeiten von Spuren mit N_{Spuren} > minimaler Wert für Untergrund- und physikalische Ereignisse ist aufgetragen. Die physikalisch relevanten Ereignisse stammen aus folgenden Ereigniskassen:

1. C.C. Ereignisse mit einem Impulsübertrag $Q^2 > 100\frac{GeV^2}{c^2}$, einer Jetenergie von $E_0 < 30GeV$ und einem Jetwinkel von $\Theta_{jet} < 0,15rad$.

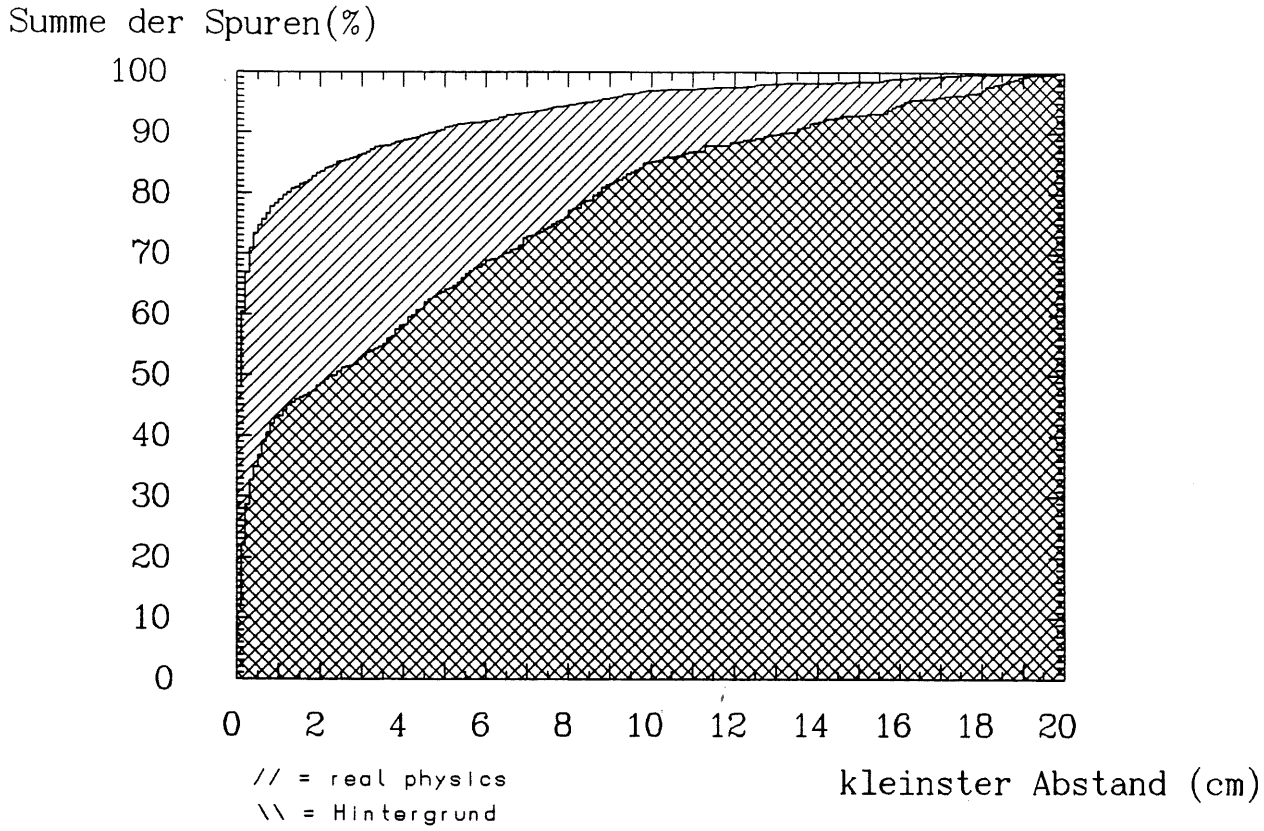


Abbildung 33: Kleinste Abstände der Spuren vom Wechselwirkungspunkt

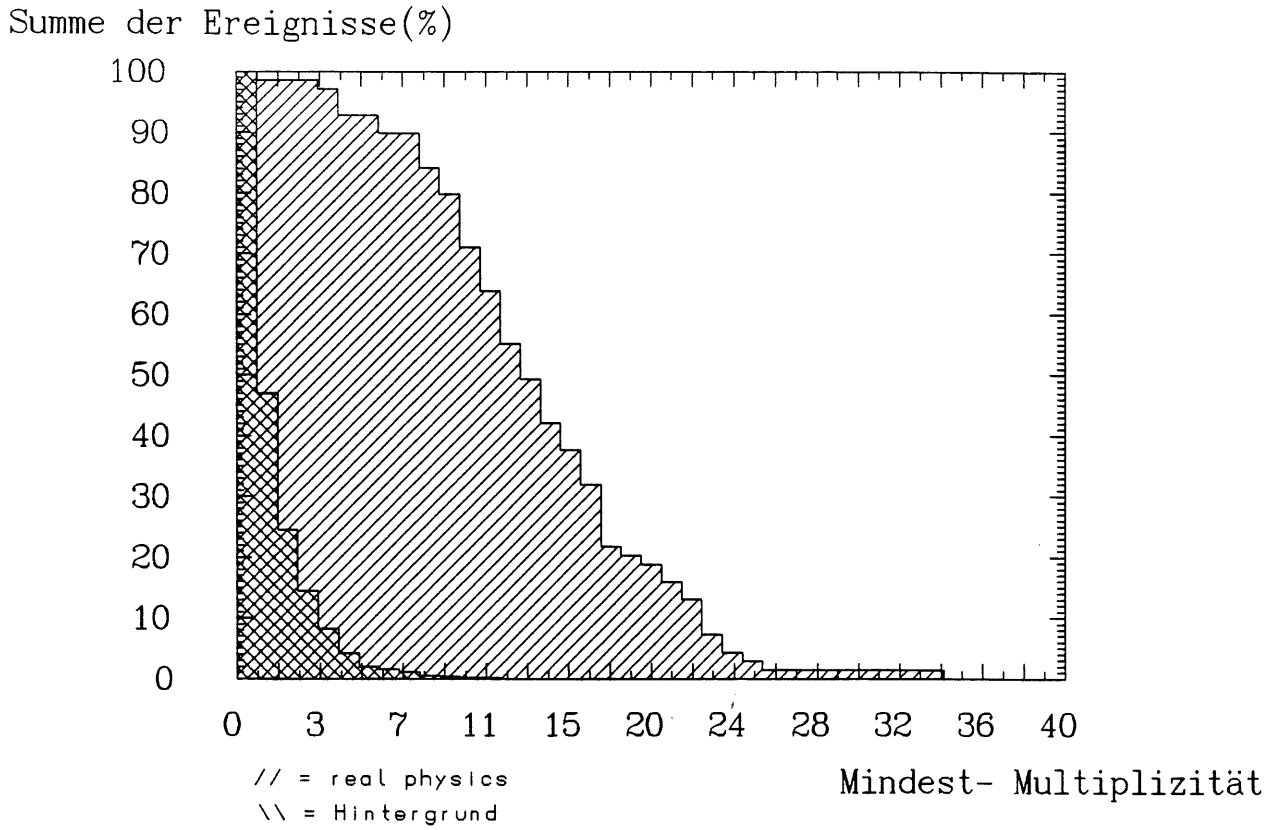


Abbildung 34: Auftretende Spurmultiplizität in den Monte–Carlo Daten

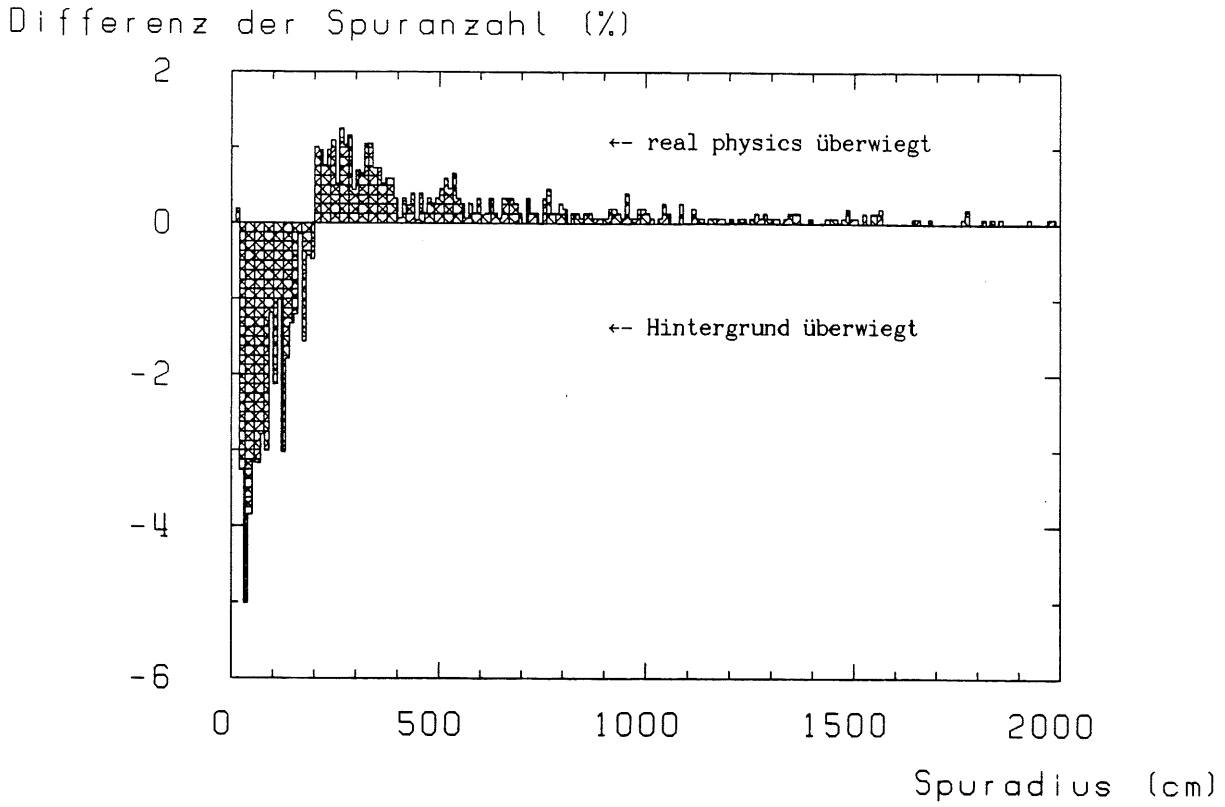


Abbildung 35: Auftretende Radien der Teilchen in den Monte–Carlo Daten

2. C.C. Ereignisse mit einem Impulsübertrag $300\frac{GeV^2}{c^2} > Q^2 > 100\frac{GeV^2}{c^2}$.

3. C.C. Ereignisse mit einem Impulsübertrag $300\frac{GeV^2}{c^2} > Q^2 > 100\frac{GeV^2}{c^2}$ und einem x-Faktor von $0,003 < x < 0,02$.

Die gefundene Multiplizität bei Ereignissen des Untergrundes ist in 92% der Fälle kleiner als 2 Spuren. Die meisten Untergrundereignisse hatten keine vollständige Spur in der CJC. Weniger als 1% der physikalischen Ereignisse besaßen dagegen eine Multiplizität von weniger als 2 Spuren. Somit ist dieser Parameter der geeigneteste, um eine wirkungsvolle Reduktion des Untergrundes zu erreichen.

In Bild 35 ist das Ergebnis der Untersuchung der auftretenden Transversalimpulse geladener Teilchen für den Untergrund und den *real physics* dargestellt. Es wurde die Differenz der Häufigkeiten von Spuren mit entsprechenden Spurkrümmungen bei Untergrundereignissen und *real physics* Ereignissen dargestellt. Bei der Untersuchung der auftretenden Transversalimpulse der Spuren bei Untergrund- und

Parameter	Simulation			Einheiten
	A	B	C	
$d_{Vertex\ Abstand}$	0,2	0.2	0,2	cm
minimaler transversaler Impuls	630	630	380	$\frac{MeV}{c}$
minimale Multiplizität	6	4	6	Spuren

Tabelle 5: Gewählte Parameter für die Simulation

real physics Ereignissen fiel auf, daß große minimale Impulse für die Rekonstruktion der Spuren gewählt werden sollten, um den Untergrund gegenüber den *real physics* Ereignissen stärker zu reduzieren. Da jedoch die Multiplizität der Ereignisse sich als ein wirksameres Mittel zur Reduktion heraus gestellt hatte, und dessen Funktionstüchtigkeit durch eine zu große Wahl des minimalen Transversalimpulses stark eingeschränkt wurde, wurde ein minimaler Impuls in der Größenordnung von einigen $100\frac{MeV}{c}$ ($\hat{=}$ einen Meter Spurradius) gewählt.

7.3. Die Parameter der Simulation

Das Triggerprogramm erlaubt es drei Parameter frei zu wählen:

- Der kleinste Abstand (in der R–Φ–Ebene) rekonstruierter Spuren vom Wechselwirkungspunkt (nachfolgend $d_{VertexAbstand}$ genannt).
- Der kleinste transversale Impuls (P_T) rekonstruierter Spuren.
- Die kleinste Multiplizität der Ereignisse für eine positive Triggerentscheidung

In der Tabelle 5 sind die gewählten Parameter dargestellt. Es wurden drei Parametersätze getestet, welche mit den Buchstaben A, B und C gekennzeichnet wurden. Diese Parametersätze wurden so gewählt, so daß der simulierte Trigger eine Rate der nicht unterdrückten Untergrundereignisse von ca. 100Hz zeigte.

Diese Werte in der Tabelle 5 sind keine errechneten Werte, sondern sind durch den Vergleich der in verschiedenen Ereignissen enthaltenden Spuren und den Spuren, welche das untersuchte Triggersystem bei verschiedenen gewählten Konstanten rekonstruierte, gemessen worden. Es wurden hierfür 200 Untergrundereignisse und 80 physikalisch relevante Ereignisse zufällig ausgewählt. In den Abbildungen 36, 37 und 38 sind die Eigenschaften der Spuren dargestellt, welche von dem untersuchten Triggersystem rekonstruiert werden konnten.

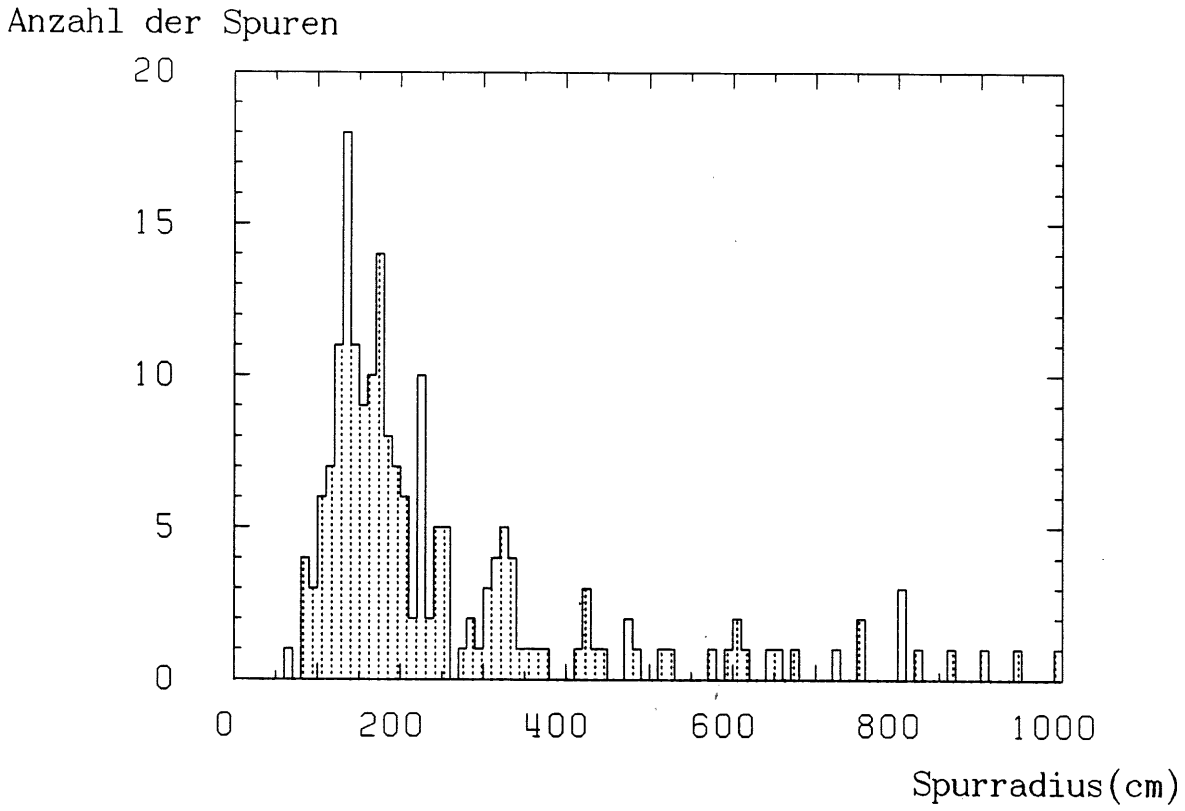


Abbildung 36: Radius der rekonstruierten Spuren (Fall A,B)

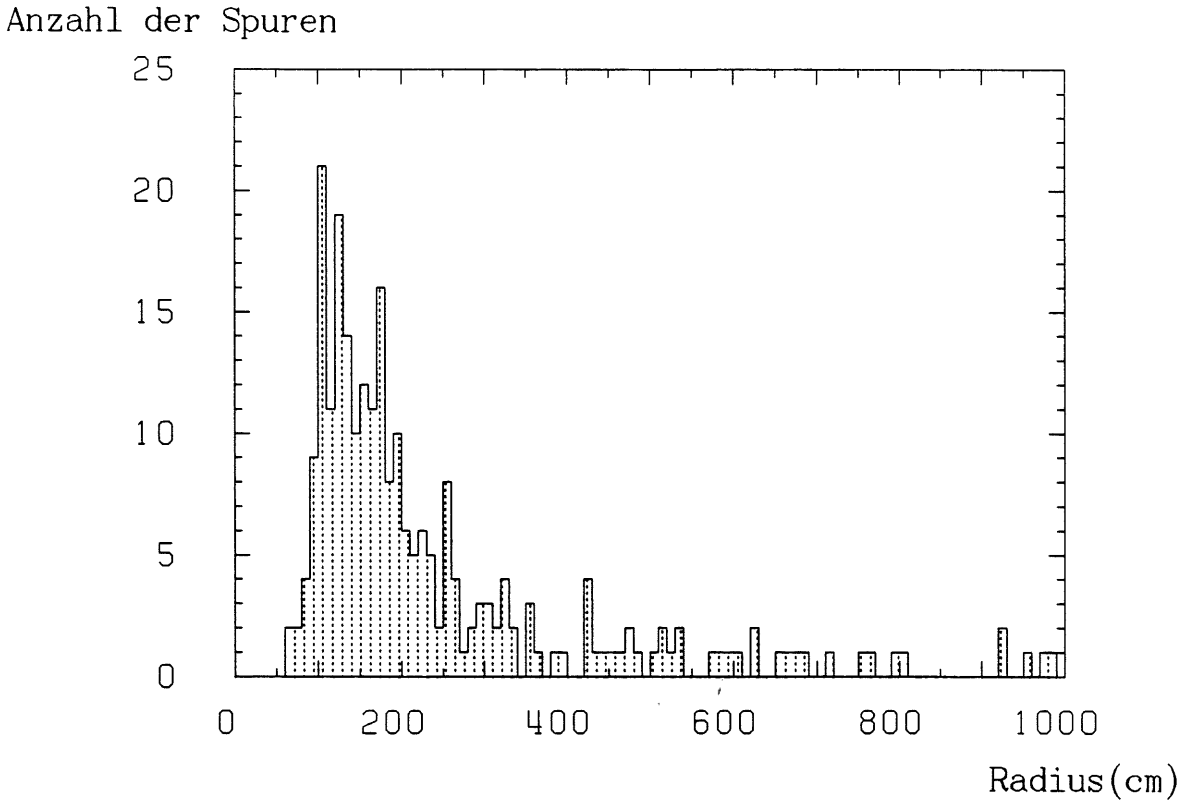


Abbildung 37: Radius der rekonstruierten Spuren (Fall C)

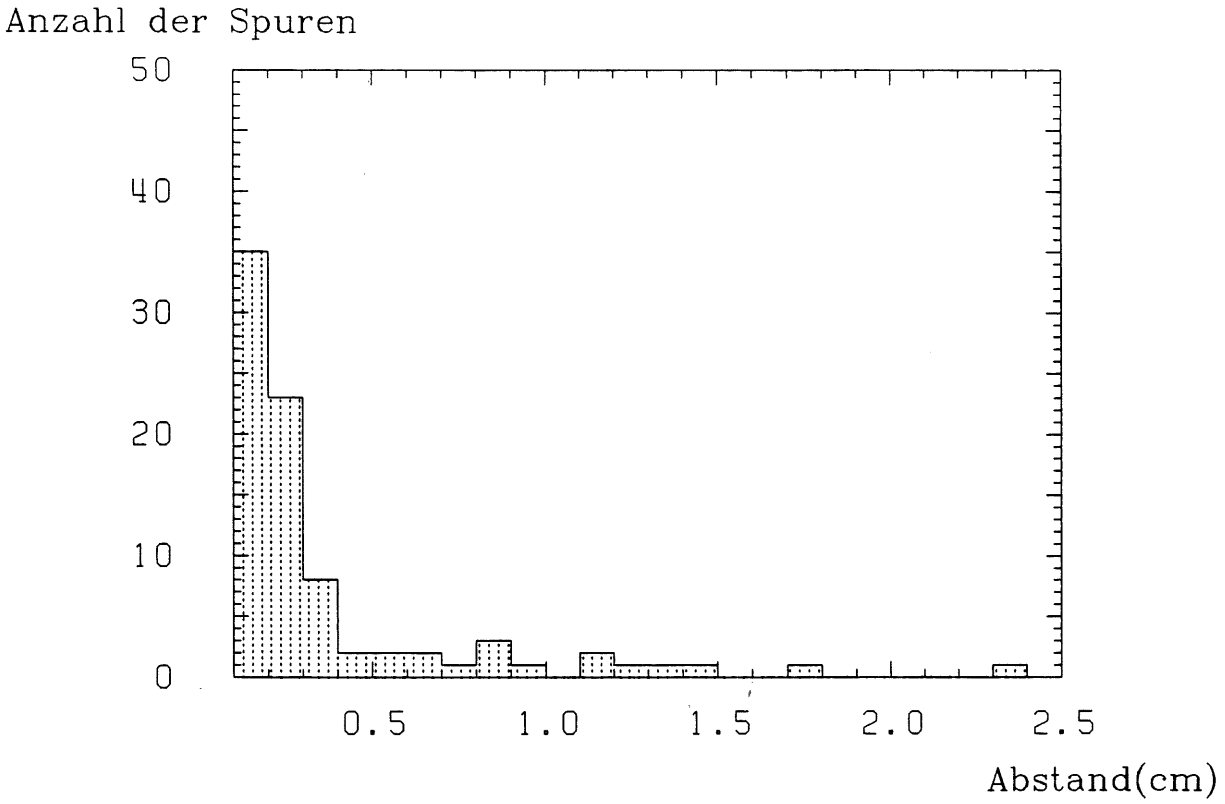


Abbildung 38: Kleinster Abstand zum Wechselwirkungspunkt (Fall A,B,C)

7.4. Simulationsablauf

In der Abbildung 39 ist der Ablauf einer Simulation des vorgestellten Triggersystemes dargestellt. Ein FORTRAN- Programm (F36WEN.HERTRAC.S(GEANT)) auf der IBM 3084Q las die unformatierten Monte- Carlo- Dateien und wandelte diese in formatierte Dateien um, da eine Übertragung von Daten über DESY-NET mit Hilfe des Übertragungsprogrammes KERMIT 2.29C bei formatierten Dateien kontrollierbarer ist. Mit Hilfe von Kermit 2.29C wurden diese Dateien über DESY-NET auf den IBM-AT übertragen. Es handelte sich in der Endphase der Simulation um einen IBM- AT kompatibler Rechner von COPAM (PC501), da dieser eine größere Rechengeschwindigkeit besaß als ein IBM- AT, bedingt durch eine höhere Taktrate (10MHz) und einen schnelleren Speicher (0 Waitstates). Als Massenspeicher wurde eine 40MByte Festplatte verwendet.

Das Programm KERMIT konvertiert die formatierten Dateien der IBM 3084Q, welche im EBCDIC- Format kodiert sind, in das ASCII- Format und überträgt diese auf den Zielrechner. Diese Dateien werden von dem in Pascal geschriebenen Programm EVENT4A.COM gelesen. Dieses Programm erzeugt für jedes Ereignis eine Ladedatei für den Simulator. Es wurde Pascal als Programmiersprache gewählt, da der TURBO Pascal Compiler von Borland in der Version 3.0 einer der schnellsten Programm Entwicklungsumgebungen auf einem IBM- Personalcomputer darstellt.

Ein weiteres Programm (COMAND.COM) erzeugt nun eine Stapeldatei (englisch: batch-file), welche die gesamte Simulation steuert, wozu der Ladebefehl der erzeugten Ladedateien gehört, das Laden der Tabellen und des Programmes, das Setzen der Zähler und das Starten der Simulation. Diese Stapeldatei wird von dem Simulator gelesen und abgearbeitet. Es wurde als Name für diese Stapeldatei RUN.CMD gewählt.

Eine Stapeldatei (WVW.BAT) initialisiert den COPAM- Rechner durch Neuladen des Betriebssystemes und des Simulators. Dann übergibt diese Datei die Steuerung an RUN.CMD

Nachdem die Simulation abgeschlossen ist, hängt das Pascal- Programm MERGED.COM alle einzelne Ergebnisdateien (Ereignisname.LOG) zusammen, um Speicherplatz auf der Festplatte zu sparen. Diese zusammengehängte Datei (Ereignisklasse.SUM) wird nun von dem Programm AUSWERT.COM, welches ebenfalls in Pascal geschrieben wurde, ausgewertet, indem eine Datei (Ereignisklasse.ASW) erzeugt wird, welche die Zahl der in der Simulation rekonstruierten Spuren, deren Referenzwinkel, den errechneten Radius und die benötigte Simulationszeit in Zyklen enthält.

Da diese Datei formatiert ist, kann sie auf die IBM 3084Q zurück übertragen werden. Das Programm HISTOGR, REDUCT, CUTS und VERTEX erzeugen Graphische Dateien (GEP, englisch: graphical editing programm), welche die Ergebnisse der Simulation und die Eigenschaften der Ereignisse, die in den Monte- Carlo Dateien enthalten sind, in Histogrammen darstellen. Der Anhang IV.1. enthält ein Beispiel einer solchen Simulation.

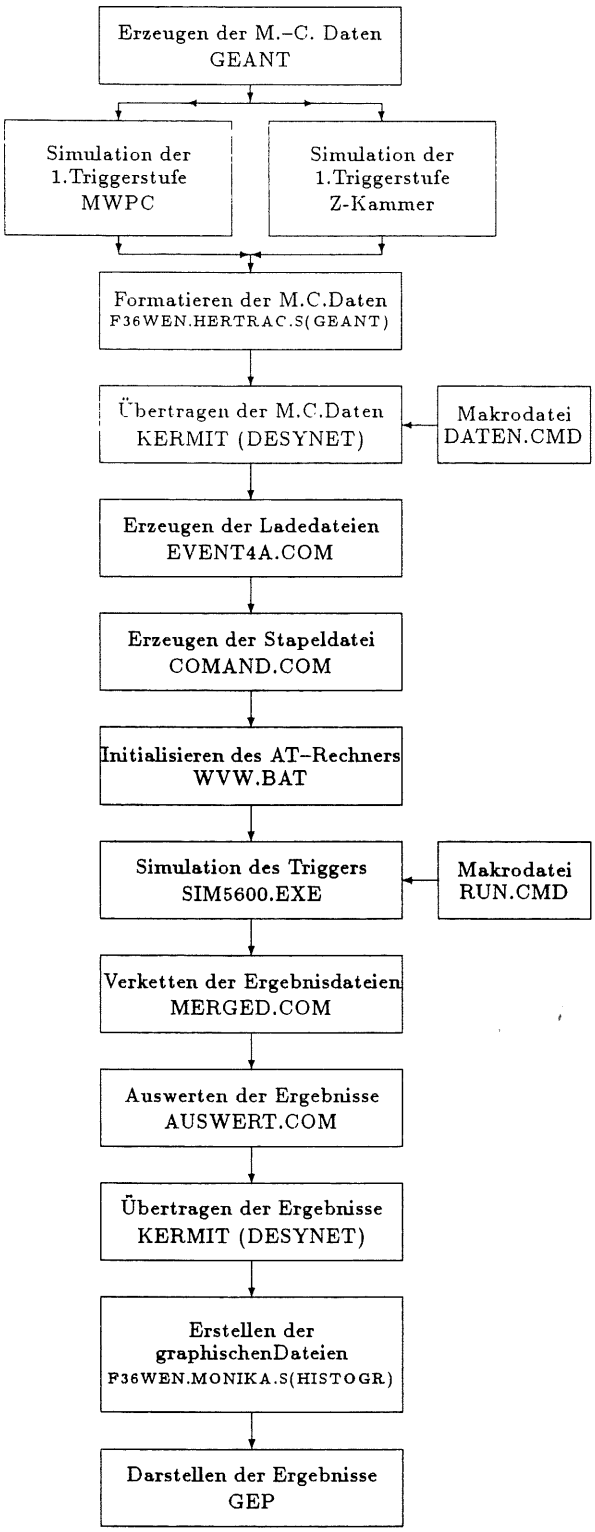


Abbildung 39: Zeitlicher Ablauf einer Simulation

7.5. Die Ergebnisse der Simulation

Die wichtigsten Ergebnisse der Simulation sind:

Zeitbedarf: Der vorgestellte Trigger erfüllt die Anforderung ein Triggersignal in maximal **0,5msec** zu liefern!
Die für eine Entscheidung "*schlechtes Ereignis*" benötigte **mittlere Zeit** beträgt nur ca **0,1msec**. Als **maximale Zeit** für die Entscheidung ein Ereignis **zu unterdrücken** wurde **0,5msec** beobachtet. Die maximale Zeit für die Entscheidung ein Ereignis **nicht zu unterdrücken** betrug 0,9msec. Diese 0,9msec reichen aus, da diese Art von Ereignisse weiter verarbeitet werden, und für das Laden eines guten Ereignisses eine Zeit von 0,8msec benötigt wird.

Die Ereignisrate: Von den 3000 untersuchten Hintergrundereignissen (Monte–Carlo) wurde nur **ein Ereignis nicht unterdrückt**. Dies entspricht einer **Ereignisrate** von **100Hz** und eine **Reduktion des Hintergrundes** von $99,93 \pm 0,03\%$. Somit erfüllt auch in dieser Hinsicht der untersuchte Trigger die Anforderung. Der minimale Transversalimpuls rekonstruierter Spuren betrug einige MeV. Die Mindestmultiplizität wurde auf 4–6 Spuren festgelegt.

Die physikalischen Ereignisse: Da der vorgestellte Trigger nur die Daten der zentralen Driftkammer verwendet, werden im Mittel ca. **30%** der "**real physics**"- Ereignisse **unterdrückt**. Diese Ereignisse haben einen so großen Vorwärtsimpuls, so daß die Teilchen die zentrale Driftkammer zu früh verlassen. Physikalische Ereignisse mit mehr als drei vollständigen Spuren in der zentralen Driftkammer wurden zu $100 \pm 2\%$ getriggert.

Parameter	Simulation				Einheiten
	A	A*	B	C	
$d_{Vertex\ Abstand}$	0,2	0,2	0,2	0,2	cm
minimaler transversaler Impuls	630	630	630	380	$\frac{MeV}{c}$
minimale Multiplizität	6	6	4	6	Spuren
Wird die Winkelinformation des Vorwärts–MWPC verwendet?	Nein	Ja	Nein	Nein	
mittlere Zeit für negative Triggerentscheidung	74±89	41±64	93±77	90±89	μsec
maximale Zeit für negative Triggerentscheidung	450	375	480	490	μsec
Zahl der getriggerten Untergrundereignisse	1	1	3	1	Ereignisse
Reduktion des Untergrundes	99,93% ±0,03%	99,93% ±0,03%	99,91% ±0,01%	99,93% ±0,03%	Ereignisse
Reduktion der physikalischen Ereignisse mit mehr als der Mindestspuranzahl in der CJC	0 ± 1%	0 ± 1%	0 ± 1%	0 ± 1%	Ereignisse
mittlere gefundene Multiplizität (größer 0) des Untergrundes	2,0±1,3	2,0±1,3	2,0±1,3	1,8±1,3	Spuren
mittlere gefundene Multiplizität der (größer 0) physikalischen Ereignisse	6,8±4,4	6,8±4,4	6,8±4,4	6,9±4,5	Spuren
mittlere gefundene Multiplizität bei Ereignissen mit 10 'geometrischen' Spuren	9,3	9,3	9,3	9,7	Spuren
maximale Zeit bis Programmende	980	—	980	970	μsec

Tabelle 6: Ergebnisse der Simulation 1.Teil

Parameter	Simulation				Einheiten
	A	A*	B	C	
$d_{Vertex\ Abstand}$	0,2	0,2	0,2	0,2	cm
minimaler transversaler Impuls	630	630	630	380	$\frac{MeV}{c}$
minimale Multiplizität	6	6	4	6	Spuren
Wird die Winkelinformation des Vorwärts–MWPC verwendet?	Nein	Ja	Nein	Nein	
Reduktion der $100 < Q^2 < 300 \frac{GeV^2}{c^2}$ $0{,}003 < x < 0{,}02$ C.C Ereignisse	$44 \pm 1\%$	$44 \pm 1\%$	$11 \pm 1\%$	$11 \pm 1\%$	Ereignisse
Reduktion der $Q^2 > 1000 \frac{GeV^2}{c^2}$ $0{,}26 > \Theta_{jet} > 0{,}16rad$ oder $\Theta_{jet} > 0{,}52rad$ N.C. Ereignisse	$48 \pm 1\%$	$48 \pm 1\%$	$45 \pm 1\%$	$45 \pm 1\%$	Ereignisse
Reduktion der $Q^2 > 1000 \frac{GeV^2}{c^2}$ $\Theta_{jet} < 0{,}15rad\ E_0 < 30GeV$ N.C. Ereignisse	$23 \pm 1\%$	$23 \pm 1\%$	$13 \pm 1\%$	$19 \pm 1\%$	Ereignisse
Reduktion der $t\bar{t}$ Ereignisse $Q^2 > 1000 \frac{GeV^2}{c^2}$ $Masse_{Top-Quark} = 40 \frac{Gev}{c^2}$	$40 \pm 1\%$	$40 \pm 1\%$	$40 \pm 1\%$	$40 \pm 1\%$	Ereignisse
Reduktion der $300 > Q^2 > 1000 \frac{GeV^2}{c^2}$ C.C. Ereignisse	$31 \pm 1\%$	$31 \pm 1\%$	$25 \pm 1\%$	$25 \pm 1\%$	Ereignisse
Reduktion der $Q^2 > 1000 \frac{GeV^2}{c^2}$ N.C. Ereignisse	$62 \pm 1\%$	$62 \pm 1\%$	$48 \pm 1\%$	$57 \pm 1\%$	Ereignisse
Mittlere Reduktion der physikalischen Ereignisse (alle zusammen)	$39 \pm 1\%$	$39 \pm 1\%$	$31{,}0 \pm 1\%$	$34 \pm 1\%$	Ereignisse
Zahl der getriggerten Ereignisse (alle zusammen)	68	68	76	73	Ereignisse
Zahl der nicht getriggerten physikalischen Ereignisse (alle zusammen)	43	43	35	38	Ereignisse

Tabelle 7: Ergebnisse der Simulation 2.Teil

Die vollständigen Ergebnisse der Simulation sind in der Tabelle 6 und 7 dargestellt. Die Simulation A* benutzt dieselben Parameter wie die Simulation A. Bei A* jedoch wurden nur die Signale in der Simulation verwendet, welche in den Winkelsektoren lagen, in denen der Vorwärts–MWPC– Trigger Spuren gefunden hat. Bei allen anderen Simulationen wurde diese Information nicht weiter verwendet. Die angegebene Zeiten sind abgeschätzte Maximalzeiten für die Version des DSP56001, welche im Frühjahr lieferbar sein soll. Für die zur Zeit lieferbare Version müssen die Zeiten mit dem Faktor 2 multipliziert werden. In den Abbildungen 40, 41 und 42 sind die benötigten Zeiten zur Erzeugung des Signales "*nicht Getriggert*" für die Untergrundereignisse bei den verschiedenen Parametersätzen (Fälle) dargestellt. Für die angegebenen Zeiten gilt dasselbe wie für die Tabelle 6 und 7. Es zeigt sich in diesen drei Abbildungen, daß die Laufzeit des Programmes nur begrenzt von den gewählten Parametern abhängt. Lediglich die geforderte Mindestmultiplizität hat einen Einfluß, da bei dem Vergrößern dieses Wertes, der Trigger mehr Spuren Rekonstruieren muß, bevor eine Entscheidung gefällt werden kann. Auffällig ist, daß bei allen drei Abbildungen ein großer Teil der unterdrückten Ereignisse eine Laufzeit von einigen msec zeigten. Dies hängt mit der Tatsache zusammen, daß das vorgestellte Programm als erstes die Anzahl der Signale in der Referenzkammer prüfte. Falls diese unter der Mindestmultiplizität lag, brach das Programm sofort ab.

Die nicht getriggerten physikalisch relevanten Ereignisse haben in der zentralen Driftkammer eine zu kleine Multiplizität. In Abbildung 43 ist ein solches Ereignis in der R–Φ Ebene dargestellt. Wie man an der einen hochenergetischen einzelnen Spur erkennen kann, handelt es sich hierbei um ein CC Ereignis. Die Jetenergie E_{jet} war kleiner 30GeV und der Jetwinkel kleiner gleich 0,15rad.

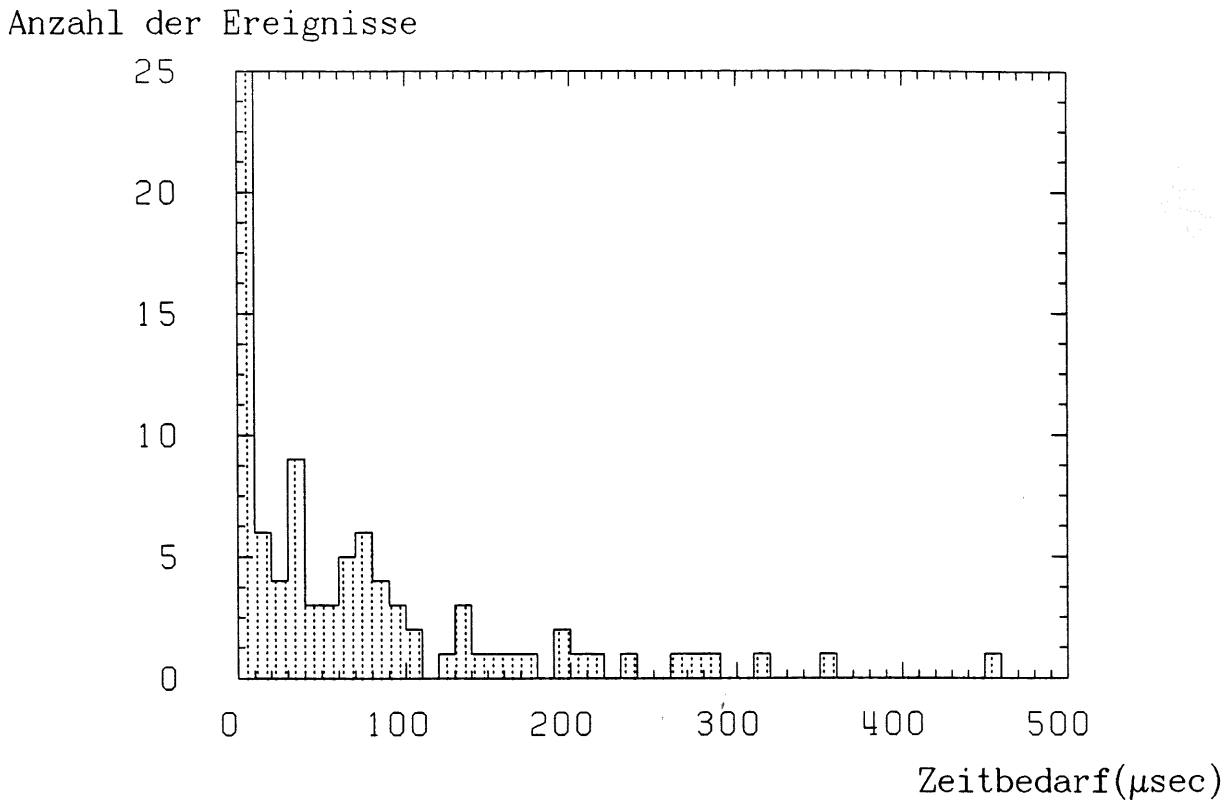


Abbildung 40: Programmlaufzeiten für Untergrundereignisse (Fall A)

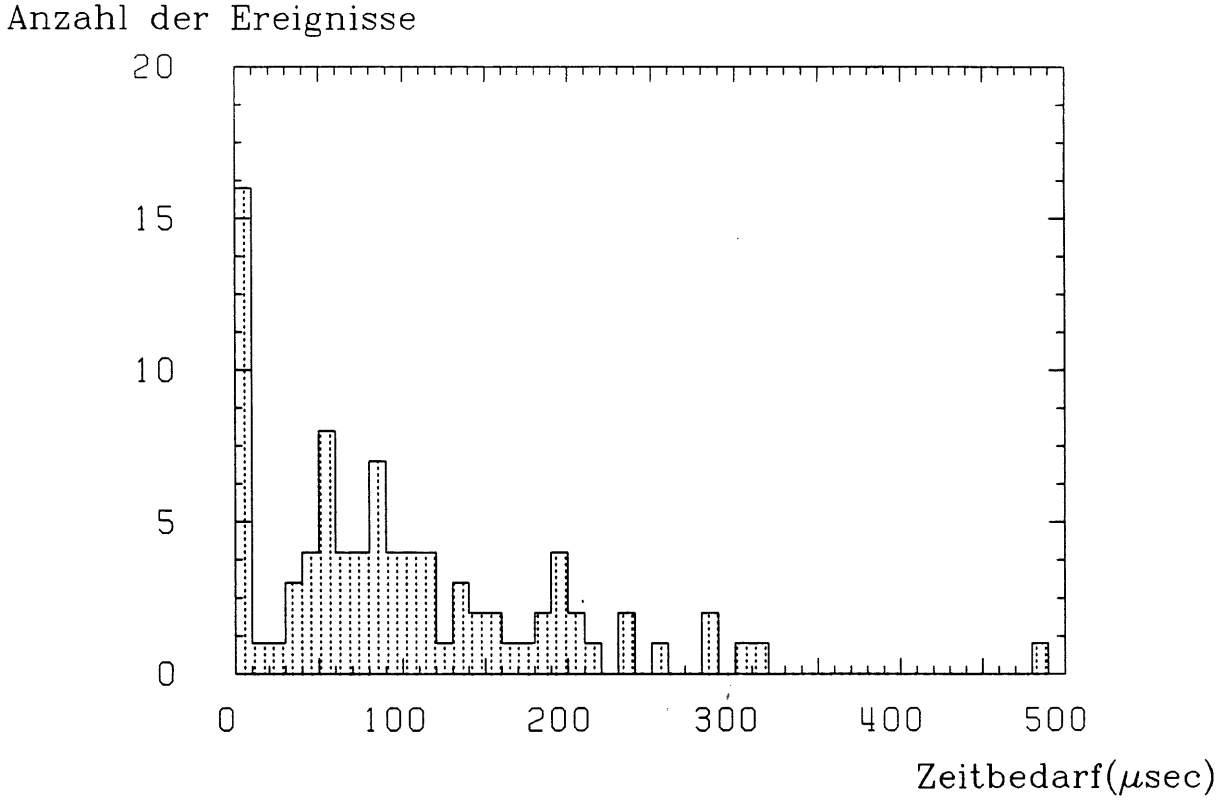


Abbildung 41: Programmlaufzeiten für Untergrundereignisse (Fall B)

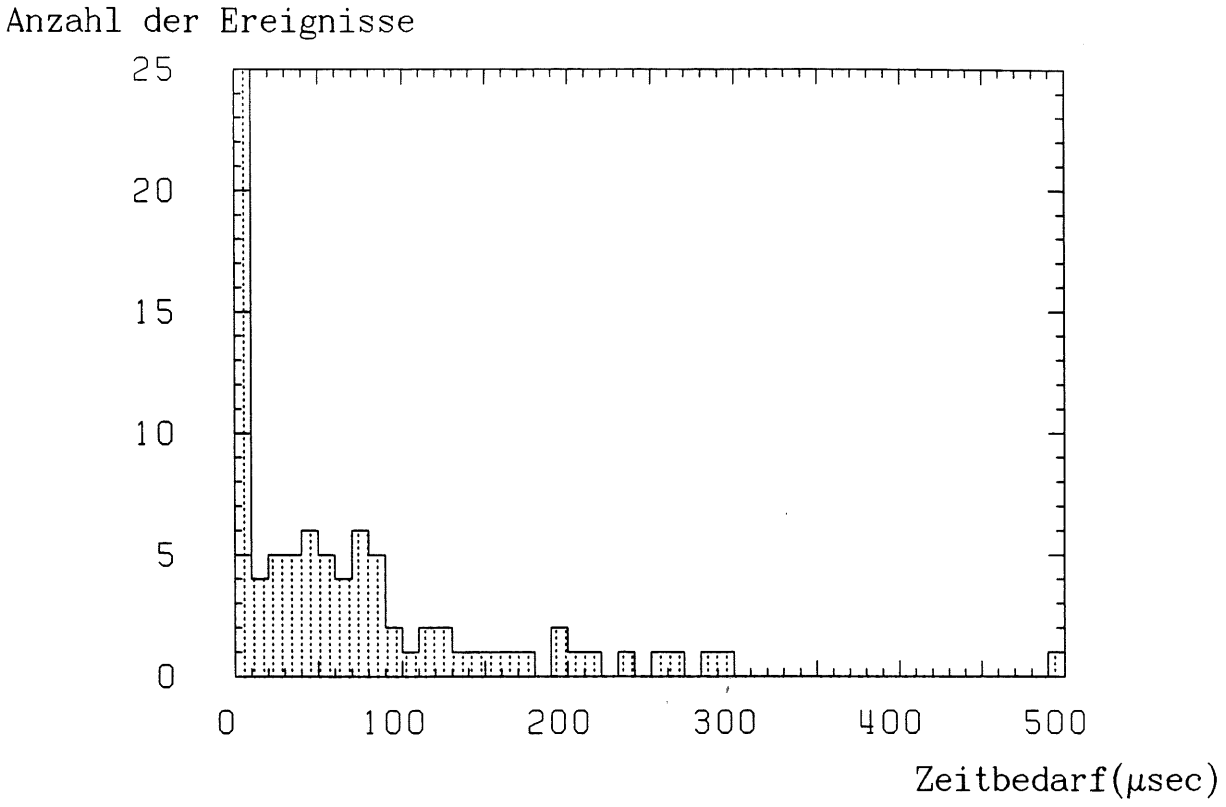


Abbildung 42: Programmlaufzeiten für Untergrundereignisse (Fall C)

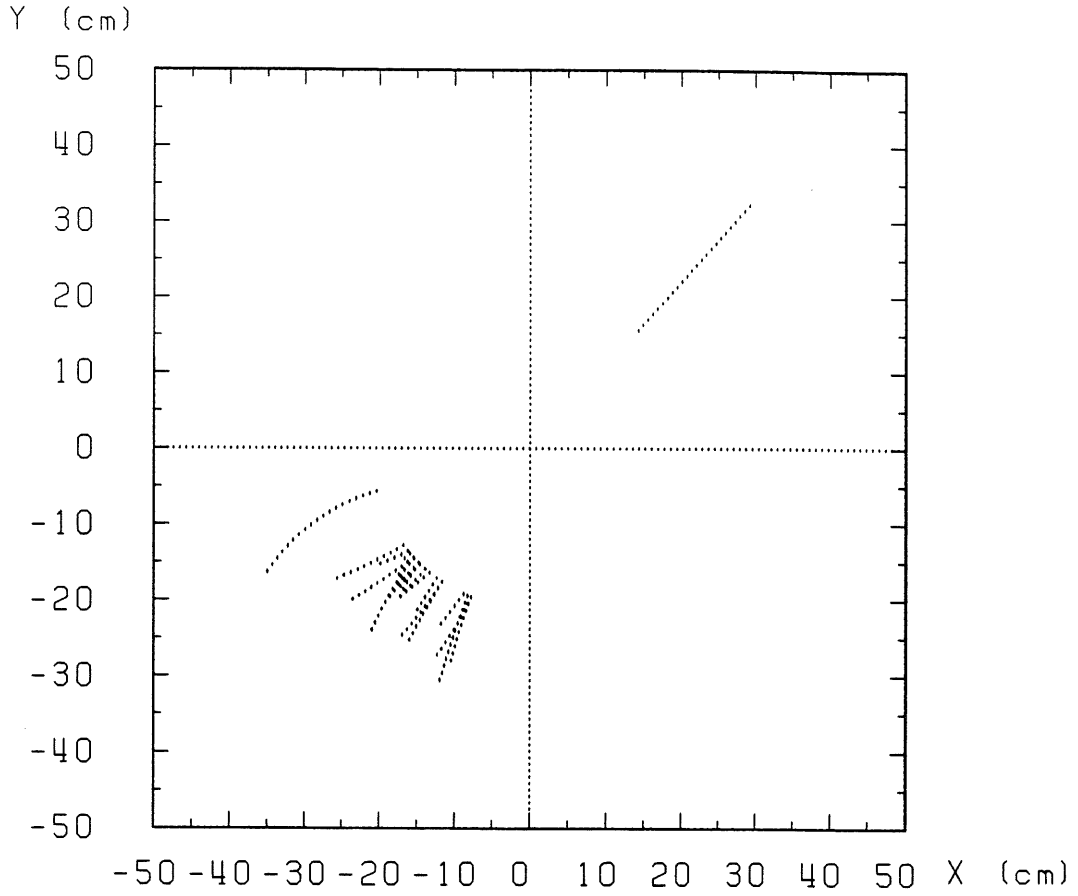


Abbildung 43: Ein nicht getriggertes *real physics* Ereignis

8 Die Kosten eines solchen Triggers

Die Kosten eines solchen Systemes können unterteilt werden in die Kosten fünf verschiedener Komponenten:

- Die Prozessorkomponente mit dem digitalen Signal–Prozessor
- Die Speicherkomponenten mit den verwendeten CAM's
- Die Schieberegister–Komponente, welche die Daten der zentralen Driftkammer synchronisiert und zwischenspeichert.
- Die Winkelberechnungskomponente, welche aus den Schieberegisterdaten die zugehörigen Winkel berechnen.
- Der VME–Einbaurahmen mit Spannungsversorgung und dem VME–Bus

Die Preise für die einzelnen Komponenten hängen stark von der Art der Entwicklung der Leiterplatten (Platinen) ab. (Werden diese bei DESY entwickelt, oder werden Fremdfirmen dazu eingesetzt?) Da die Kosten für eine Eigenentwicklung bei DESY schlecht anzugeben sind, entschied ich mich bei der Abschätzung der Preise für die Kosten bei Fremdfirmen.

8.1. Die Kosten der Prozessorkarte

Die Prozessorkarte besteht aus dem DSP56001 Prozessor von Motorola, den Festwertspeichern für die Tabellen und dem Programm, einer VME- Schnittstelle und den Systemfunktionen wie z.B. der Takterzeugung. Auf Grund der hohen Frequenzen und der hohen Anschlußdichte des DSP56001 werden Leiterplatten mit vier Leiterbahnebenen benötigt (englisch: four layer planes). Die Enwicklung einer solchen Leiterplatte kostet bei einer Fremdfirma ca.25000DM. Die Kosten für eine Platine mit allen Komponenten wie Stecker und elektronischen Bauteilen kosten dann zusätzlich ca. 3000DM. Die Industrie bietet bereits komplette DSP56001-Systeme für VME an. Solche Systeme kosten ca. 10000DM. Es wäre im Falle des Aufbaus eines solchen Triggersystemes zu prüfen, ob solche Platinen bereits alle benötigten Funktionen besitzen. Alle Preise sind in Tabelle 8 aufgeführt.

8.2. Die Kosten der CAM–Karten

In der Simulation wurden für jede Signaldrahtebene 40 Speicherstellen für Signale reserviert. Jede dieser Speicherstellen hatte eine Wortbreite von 10Bit. Somit werden $23 \times 40 \times 10$ Speicherbits in CAMS benötigt. Die lieferbaren CAMS enthalten 16 Speicherstellen pro Baustein. Diese können verschieden organisiert sein ($8 \times 2, 4 \times 4$). Somit werden mindestens 575 Bausteine benötigt. Da die Kosten der Bausteine bei ca. 30DM liegen, kosten alle CAM's zusammen ca. 18000DM. Für die 575 Bausteine

Baugruppe	Einzelpreis (DM)	benötigte Menge	Preis (DM)
Entwicklungskosten für Platine (Layout)	25000	1	25000
Kosten für Bauteile, Platine und Stecker	3000	1	3000
Kosten für industrielle Prozessorkarte	10000	1	10000

Tabelle 8: Kosten der Prozessorkarte

Baugruppe	Einzelpreis (DM)	benötigte Menge	Preis (DM)
Entwicklungskosten für Platine (Layout)	25000	1	25000
Kosten für Bauteile, Platine und Stecker	1000	12	12000
Kosten für CAM–Bausteine	30	575	18000

Tabelle 9: Kosten des CAM–Speichers

werden 12 Leiterplatten incl. der VME–Steuerung benötigt. Auch diese Leiterplatten müssen auf Grund der großen Zahl von enthaltenen Bausteinen und der hohen Taktraten mindesten vier Lagen Leiterbahnen besitzen. Die Kosten zur Entwicklung dieser Platine beträgt ca. 25000DM. Jede einzelne Platine kostet mit den Bausteinen (ohne CAMS) und Steckern 1000DM. Zusammen betragen die Kosten für die CAM–Karten ca. 55000DM (siehe Tabelle 9).

8.3. Die Kosten der Schieberegister–Karten

Die Kosten der Karten mit den Schieberegister zum Zwischenspeichern der Signale der zentralen Driftkammer belaufen sich auf 18500DM. Diese Karten enthalten im wesentlichen die Schieberegister und Synchronisiereinheiten für die Daten. Da das

Baugruppe	Einzelpreis (DM)	benötigte Menge	Preis (DM)
Entwicklungskosten für Platine (Layout)	12500	1	12500
Kosten für Bauteile, Platine und Stecker	1000	6	6000

Tabelle 10: Kosten der Schieberegister–Karten

II. Institut für Experimentalphysik der Universität Hamburg für den Bau der Myon–Trigger eigene Bausteine (Gatearray Plessay CLA 53024) entwickelt haben, welche die benötigten Komponenten für 8 Kanäle enthalten, bestehen die Schieberegisterplatinen aus 90 dieser Bausteinen und 720 Steckern für die Verbindungen zur zentralen Jetkammer. Da die 720 Verbindungen zu den "Flash ADC" Auslesekarten der zentralen Driftkammer eine Distanz von mindesten 2m überbrücken, sind zusätzliche Masse–Kabel zur Abschirmung notwendig. Somit ergeben sich 1440 Stecker. Eine so große Zahl an Steckern sind pro VME–Karte nicht möglich. Daher werden sechs dieser Karten mit jeweils ca 15 Bausteine und 240 Stecker benötigt. Da die Integrationsdichte diese Karten nicht sehr hoch ist, muß die Platine nur zwei Lagen Leiterbahnen besitzen. Die Entwicklungskosten solcher Platinen liegen zwischen 10000–15000DM. Ich nehme für die weitere Berechnung ein Preis von 12500DM an. Die Kosten für jede einzelne Platine mit Steckern und den Bausteinen (Gatearrays) betragen dann nochmal 1000DM.

8.4. Die Kosten der Winkelberechnungseinheit

Diese Einheit muß die Signale der zentralen Driftkammer in Winkelwerte umrechnen. Solche Systeme wurden bei diversen Experimente bei DESY in der Vergangenheit bereits eingesetzt (z.B. TASSO). Die Preise solcher Systeme belaufen sich auf 1000DM. Sie enthalten im wesentlichen schnelle Addierer und Multiplizierer und kleine Festwertspeicher mit Tabellen. Die Platine kann zweilagig aufgebaut sein, so daß nochmals 10000–15000DM für die Platinenentwicklung dazu käme.

8.5. Die Gesamtkosten

Die einzelnen Karten werden untergebracht in einem VME–Einsteckrahmen (englisch: crate), welcher auch das Netzteil und den VME–Bus enthält. Solche Rahmen kosten zwischen 3000DM und 5000DM. Sie besitzen 21 Steckplätze, wovon der vorgeschlagene Trigger maximal 20 belegt.

Baugruppe	Einzelpreis (DM)	benötigte Menge	Preis (DM)
Entwicklungskosten für Platine (Layout)	12500	1	12500
Kosten für Bauteile, Platine und Stecker	1000	1	1000

Tabelle 11: Kosten der Winkelberechnungs–Einheiten

Bei der Untersuchung, bei welchen Komponenten Einsparungen gemacht werden können, fallen besonders die hohen Kosten für die CAM–Karten auf. Die CAM–Karten könnten ersetzt werden durch spezielle RAM–Karten, welche einen geeignet strukturierten Aufbau besitzen. Der Preis solcher Karten würde zusammen bei ca. 35000DM liegen. Jedoch wäre die Zugriffsgeschwindigkeit um mindestens 50% kleiner. Eine zweite Möglichkeit wäre es, ganz auf solche speziellen Speicher zu verzichten, und normalen Schreib–Lese–Speicher zu verwenden. Dann jedoch ist die Zugriffsgeschwindigkeit je nach verwendeter Maskenbreite (4–16 Bits) um 100%–800% langsamer. Der Schreib–Lese–Speicher jedoch könnte mit auf der Prozessorkarte integriert werden. Somit entfallen die Kosten für CAM–Karten.

Bauform	Preis (DM)
System mit einer industriellen Prozessorkarte und CAM–Karten	101000
System mit einer eigenentwickelter Prozessorkarte und CAM–Karten	119000
System mit einer industriellen Prozessorkarte und spezielle RAM–Karten	81000
System mit einer industriellen Prozessorkarte und normalen Schreib–Lese–Speicher	46000
System mit zwei industriellen Prozessorkarten und normalen Schreib–Lese–Speicher	57000
System mit zwei eigenentwickelten Prozessorkarte und normalen Schreib–Lese–Speicher	68000

Tabelle 12: Gesamtkosten des Systems (verschiedene Konfigurationen)

9 Möglichkeiten zur Erweiterung eines solchen Triggers

Bisher wurde immer angenommen, daß das Prozessorsystem nur einen Prozessor besitzt, welcher die Triggerentscheidung zu bilden hat. In diesem Kapitel soll untersucht werden, ob nicht eine Verkürzung der Entscheidungszeit möglich ist durch den Einsatz mehrerer Prozessoren (englisch: multi processing). Es existieren im wesentlichen vier Konzepte zur Gestaltung solcher Multiprozessorsystemen:

1. Jeder Prozessor bearbeitet ein Ereignis. Mehrere Ereignisse werden zeitgleich von dem Multiprozessorsystem verarbeitet (Parallele Ereignisverarbeitung).
2. Jeder Prozessor bearbeitet ein Winkelsegment der Driftkammer. Es werden zeitgleich die Signale jedes Winkelsegmentes verarbeitet (Parallele Winkelsegment- Verarbeitung).
3. Jeder Prozessor bearbeitet eine Spur (ein Treffer in der Referenzkammer). Es werden zeitgleich die Spuren abgearbeitet (Parallele Spurverarbeitung).
4. Jeder Prozessor rekonstruiert nur einen Teil der Spur. Ein übergeordneter Prozessor setzt dann die Spur aus den Teilen zusammen (Parallele Verarbietung von Spursegmenten).

Jeder dieser Verfahren hat spezielle Nach- und Vorteile.

9.1. Parallele Ereignisverarbeitung

Falls jeder Prozessor ein Ereignis verarbeitet, also mehrere Ereignisse parallel vom Multiprozessorsystem verarbeitet werden, so ergeben sich folgende Vorteile:

- Jeder Prozessor hat seine eigenen Daten im Speicher. Es muß der Zugriff auf diesen Speicher nicht synchronisiert werden mit anderen Prozessoren.
- Die Leistungssteigerung eines solchen Multiprozessorsystems ist proportional zu der verwendeten Prozessoranzahl.
- Verschiedene Prozessorarten können im selben Prozessorsystem betrieben werden.
- Eine Funktionskontrolle jedes Prozessors ist relativ einfach möglich, in dem zwei Prozessoren ein und dasselbe Ereignis verarbeiten, und das Ergebnis verglichen wird.
- Die Verwaltung der einzelnen Prozessoren ist einfach, da keine Aufteilung der Zuständigkeit innerhalb eines Ereignisses existiert.
- Bei kleinen Ereignisraten ist die Totzeit eines solchen Systemes nahezu Null.

- Es existiert keine obere Grenze für die benutzte Anzahl von Prozessoren im System.

Damit jedoch ein solches System aufgebaut werden kann, muß es möglich sein, die Daten mehrerer Ereignisse in der Ausleseelektronik zwischenzuspeichern. Dies kann durch sogenanntes Memory Paging geschehen. Memory Paging bedeutet, daß der Speicher in der Ausleseelektronik mehrfach vorhanden ist. Ein Systemverwalter schaltet nach jedem Ereignis die aktuelle Speicherseite um. So kann jeder Prozessor die Daten einer Seite verarbeiten. Die Nachteile eines solchen Multiprozessorkonzeptes sind daher:

- Hohe Kosten für den mehrfach vorhandenen Speicher.
- Hoher Platz- und Strombedarf für den großen Speicher.
- Ein solches System kann nicht kooperieren mit anderen Triggern, welche nicht ebenfalls so aufgebaut sind, da die Triggerentscheidung zeitlich nach dem Auftreten eines eventuell nachfolgenden Ereignisses erst gefällt wird.

Dennoch ist eine solche Art der Bearbeitung eines Problemes durch mehrere Prozessoren eine weit verbreitete. Sie wird Verarbeitung durch *event pipelining* genannt.

9.2. Parallele Winkelsegment- Verarbeitung

Bei dieser Art der Parallelisierung der Verarbeitung der Triggerdaten ist jeder Prozessor nur für die Verarbeitung der Signale eines Winkelsegments zuständig. Die Vorteile eines solchen Systemes sind:

- Der Zugriff auf die Signaldaten muß nicht synchronisiert werden, da jeder Prozessor nur Daten eines Winkelbereiches verarbeitet.
- Der Speicherbedarf ist klein.

Da jedoch Spuren auch über solche Sektorgrenzen verlaufen können, muß eine Übergabe von teilweise rekonstruierten Spuren von einem zum anderen Prozessor möglich sein oder der Speicher an den Sektorgrenzen von jeweils beiden Prozessoren verwendbar sein. Somit sind die Nachteile eines solchen Systemes:

- Hoher Aufwand für die Synchronisierung der Prozessoren untereinander.
- Bedingt durch diesen Aufwand, kann ein solches System nur eine begrenzte Anzahl an Prozessoren besitzen.
- Bei stark gebündelten Jets kann es vorkommen, daß ein Prozessor alle Signaldaten verarbeiten muß und die anderen Prozessoren still stehen. Dadurch ist die Leistungssteigerung nicht proportional zur Prozessoranzahl.

- Die Verteilung der Daten auf die Prozessoren muß von der Ausleseelektronik verwaltet werden.
- Der langsamste Prozessor bestimmt die Geschwindigkeit des Systems.

Solche Art der parallelen Verarbeitung findet man häufig bei den Vektorrechnern, in dem jedes spezialisierter Prozessorteil eine spezielle festgelegte Aufgabe ausführt.

9.3. Parallele Spurverarbeitung

Bei dieser Art der Parallelisierung arbeitet jeder Prozessor ein Signal in der Referenzkammer (eine Spur) ab. Der Vorteil eines solchen Systemes ist:

- Alle Prozessoren können gleichmäßig ausgelastet werden.
- Die Prozessoren müssen untereinander keine Daten austauschen.
- Verschiedene Prozessorarten in einem System möglich.
- Kleiner Speicherbedarf.
- Die Leistungssteigerung steigt mit der Anzahl der Prozessoren stark an.

Da jedoch alle Prozessoren einen Zugriff zu allen Daten haben müssen, ist die Synchronisierung des Speicherzugriffes sehr schwierig und aufwendig. Eine große Anzahl von Prozessoren hat zur Folge, daß diese Synchronisierung zur einer starken Verringerung der Rechenleistung des Systemes führt. Die Nachteile eines solchen Konzeptes sind:

- Die Anzahl an Prozessoren ist sehr stark beschränkt.
- Eine sehr komplexe Synchronisierung des Speicherzugriffes ist nötig.

Ein solches Konzept der Parallelisierung liegt bei den Parallelen Computersystemen wie der "*connection machine*" vor.

9.4. Parallele Verarbeitung von Spursegmenten

Bei diesem Konzept, ist jeder Prozessor für ein Spursegment zuständig. Dies bedeutet, daß jeder Prozessor die Signale bestimmter Lagen von Signaldrähten verarbeitet. Dazu sind jedoch umfangreiche Untersuchungen nötig, um festzulegen, für welche Lagen welcher Prozessor zuständig ist, da die Auflösung und die erwartete Signaldichte von der Signallage abhängig ist. Die Vorteile eines solchen Konzeptes sind:

- In gewissen Grenzen ist die Leistungssteigerung eines solchen Systemes proportional der Prozessoranzahl.

- Keine Synchronisation des Speicherzugriffs nötig.
- Kleiner Speicherbedarf.

Da jedoch jede Spur von mehreren Prozessoren bearbeitet wird und auch aus einzelnen Spursegmenten zusammengesetzt werden muß, ist der Aufwand der Synchronisation der verschiedenen Prozessoren sehr groß. Auch ist durch die begrenzte Anzahl an Signaldrahtlagen die Zahl der Prozessoren im System stark beschränkt. Die Nachteile eines solchen Konzeptes sind somit:

- Sehr hoher Aufwand der Synchronisation der Prozessoren.
- Prozessoren müssen viele Daten untereinander austauschen.
- Die Auslastung der Prozessoren ist stark von einem Ereignis abhängig (z.B. mittlerer Vorwärtsimpuls).
- Nur kleine Prozessoranzahl möglich.
- Alle Prozessoren müssen gleich sein.
- Eine Kontrolle der Funktionsfähigkeit eines Prozessors ist kompliziert auf Grund der stark festgelegten Aufgabenbereiche.
- Jeder Prozessor hat ein anderes Programm.

9.5. Zusammenfassung

Es existieren verschiedene Möglichkeiten der Parallelisierung dieses Triggersystemes. Das verwendete Programm läßt die verschiedenen Konzepte zu, es müßte nur erweitert werden um eine eventuell notwendige Synchronisierung bestimmter Vorgänge. Diese Synchronisierung ist sicherlich auch durch eine zusätzliche Elektronik (Hardware) möglich, z.B. durch Speicher mit mehreren Zugriffsmöglichkeiten (englisch: dual ported RAM).

10 Schlußbetrachtung

In dieser Arbeit wurden verschiedene Algorithmen zur Rekonstruktion von Spuren in der zentralen Jetkammer auf die Verwendbarkeit für einen mikroprozessorgestützten Trigger der dritten Stufe untersucht. Ein solcher Trigger soll in 0.5msec durch Rekonstruktion von Teilchenspuren eine Triggerentscheidung liefern. Eine Spur–Rekonstruktion unter Verwendung der Baumstruktur stellte sich als zu zeitaufwendig heraus. Der Einsatz von eigenentwickelten Spezialprozessoren stellte sich in der Entwicklung als zu aufwendig heraus. Ein Algorithmus, welcher auf einer modifizierten Kreisgleichung beruht, erfüllte die Anforderung hinsichtlich des Entwicklungsaufwandes, des Zeitbedarfs und der Effektivität der Spur–Rekonstruktion. Des weiteren wurden drei verschiedene Prozessoren auf die Verwendbarkeit in dem oben erwähnten Trigger untersucht. Diese Prozessoren waren Vertreter sehr unterschiedlicher Prozessorkonzepte, nämlich der Prozeßrechner (MC68020), der Digitalen Signal Prozessoren (DSP56001) und der parallelen Prozessoren (Transputer IMS T 414). Der DSP56001, ein Digitaler Signal Prozessor, erwies sich als am besten geeignet bezüglich der Verarbeitungsgeschwindigkeit und des elektronischen Aufwandes. Der oben erwähnte Algorithmus wurde mit Monte–Carlo–Daten des Programmpaketes GEANT simuliert für Untergrund- und einige Klassen physikalischer Ereignisse welche bei dem Experiment H1 bei HERA erwartet werden. Die Simulation zeigte, daß ein solches Triggersystem möglich ist. Die Untergrundereignisrate von 3×10^6Hz wurde auf 1×10^3Hz reduziert. Der mittlere Zeitbedarf betrug 0.1msec für eine Triggerentscheidung. Die physikalischen Ereignisse mit mindestens vier vollständigen Spuren in der zentralen Driftkammer von H1 wurden zu 100% getriggert. Insgesamt wurden 70% der physikalischen Ereignisse getriggert. Die Kosten eines solchen Systems betragen mit der Ausleseelektronik 50–100kDM.

Anhang

I Der Digitale Signalprozessor DSP56001

I.1. Beschreibung des DSP56001

Der DSP56001 ist ein Digitaler Signalprozessor mit einer Datenbreite von 24 Bit. Die Arithmetikeinheit für Datenmanipulationen arbeitet in Festkomma- Darstellung (englisch: 24 bit fixed point precision). Das Funktionsmodell entspricht weitgehend dem einer register-orientierten Maschine. Auf dem Prozessorbaustein befinden sich vier Daten- und und drei Adressbusse. Diese verbinden die zehn funktionellen Gruppen des DSP56001. Die internen Daten- bzw. Adressbusse sind über einen Busschalter(englisch: external data or adress switch) mit dem externen Daten- bzw Adressbus verbunden (keine externe Harvard- Struktur).

Der DSP56001 kann logisch unterteilt werden in drei große Blöcke (Abbildung 44). Der erste Block besteht aus den funktionellen Gruppen, die zur Steuerung des Systemes bestimmt sind. Da wäre als erstes die Programm- Steuereinheit. Diese arbeitet die Befehle, die im Programmspeicher stehen, ab. Der Programmspeicher besteht aus einem 512Byte großen internen Schreib- Lesespeicher (englisch: random access memory, RAM), einem 32Byte Festwertspeicher (englisch: masked read only memory, masked ROM), der mit einem kleinem Programm programmiert ist, welches bei einem Neustart des DSP56001 das Benutzerprogramm in das Programm-RAM lädt (englisch: bootstrap program) und einem bis zu 64kByte großen externen Speicher. Die Programmdaten wandern über einen speziellen Programmdatenbus (P–DATEN) zur Programm- Steuereinheit. Diese Steuereinheit kann über einen globalen Datenbus (G–DATEN) Daten austauschen mit allen anderen Einheiten des Prozessors. Neben diesem Datenbus bestehen noch einzelne Steuerleitungen zu den anderen Einheiten.

Der zweite Block wird aufgebaut aus den Einheiten, welche Daten verarbeiten. Zur Datenmanipulation steht eine arithmetische Einheit (englisch: data arithmetic unit, DAU), die aus einem $24 \times 24 \rightarrow$ 48Bit(+8Bit Überlauf) Addierer und Multiplizierer besteht, zur Verfügung. Die Operanden der arithmetischen Funktionen wandern über zwei Datenbusse (X DATEN, Y DATEN) von zwei verschiedene Speicherblöcken, X- Speicher und Y- Speicher genannt, zu der DAU. Jeder dieser Speicherblöcke besteht aus 256 Byte masked ROM und 256 Byte Schreib- und Lesespeicher (englisch: random access memory, RAM). Beide Speicherblöcke können extern auf 64kByte erweitert werden. Die Adressen der benötigten Daten werden von einer seperaten arithmetischen Adresseinheit (englisch: control adress unit, CAU) berechnet. Diese adressiert den P-, X- und Y-Speicher über drei Adressbusse (P Adressbus, X Adressbus, Y Adressbus). Entsprechend dieser drei Adressbusse besteht die CAU intern aus drei arithmetischen Einheiten. So können im Idealfall drei Daten gleichzeitig adressiert und bewegt werden.

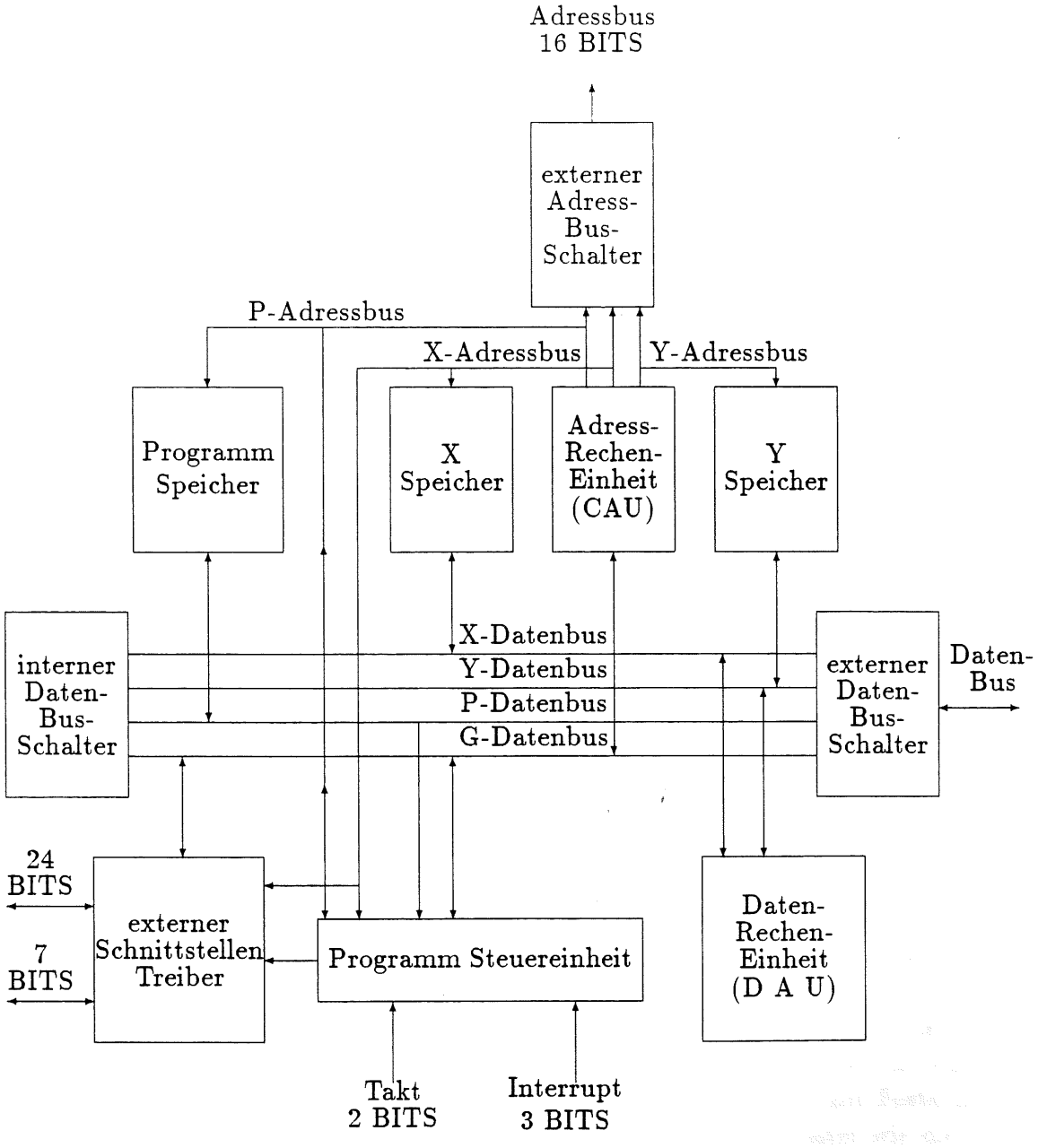


Abbildung 44: Der interne Aufbau des DSP56001 (Bockschaltbid)

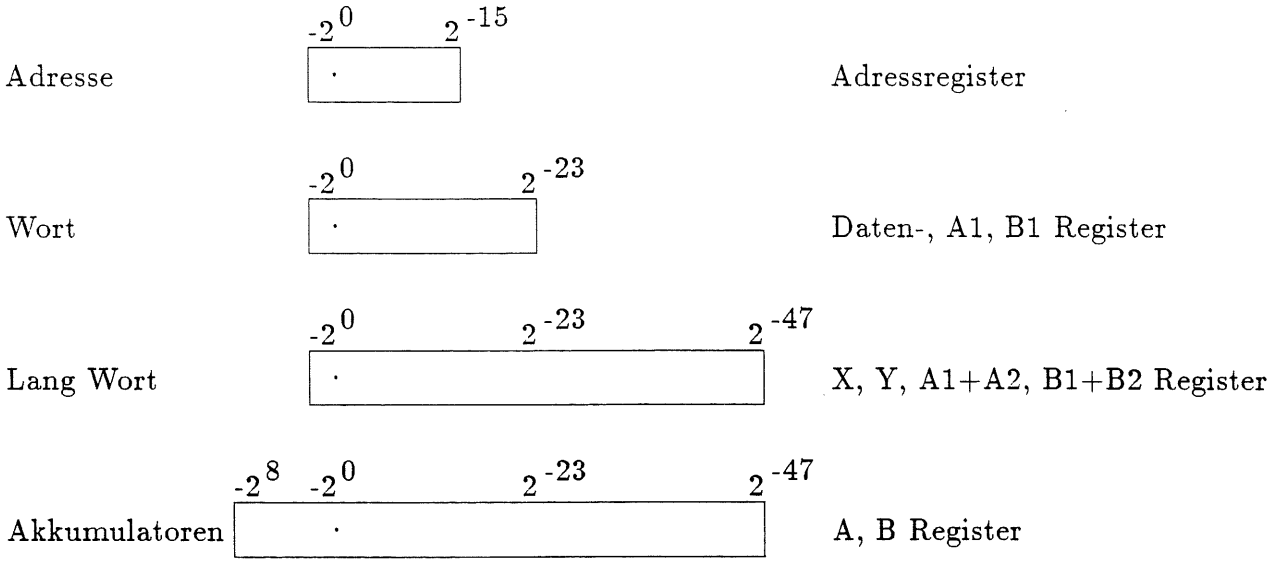


Tabelle 13: Datendarstellung in den Registern

Der dritte Block besteht aus den Schnittstellen, die sich auf dem Prozessorbaustein (englisch: chip) befinden. Die zwei seriellen und eine parallele Schnittstelle dienen der Kommunikation des Prozessors mit anderen Mikroprozessorsystemen. Die parallele Schnittstelle kann als Bus zu einem übergeordneten Leitrechner verwendet werden (englisch: host interface). Die eine serielle Schnittstelle benutzt ein synchrones (englisch: synchronous serial interface, SSI) und die andere eine asynchrones Übertragungsformat (englisch: serial communication interface, SCI). Extern stehen 64 Adressen für weitere Schnittstellen zur Verfügung. In dem Adressbereich der X- und Y- Speicherblöcke befinden sich die Kanaladressen für die Register der externen Speicherverwaltung und der Schnittstellen.

Die nach IEEE 754 genormte 32 Bit Darstellung von Fließkommazahlen (englisch: floating point number) benutzen einen 24 Bit lange Mantisse und einen 8 Bit langen Exponenten. Daher wählte die Firma Motorola eine 24 Bit Festkomma-Zahl Darstellung, da diese die gleiche numerische Genauigkeit besitzt wie die oben erwähnte floating point Darstellung. Dies setzt voraus, das die verwendeten Zahlen in der gleichen Dekade bleiben. Die Darstellung der Zahlen erfolgt in der Bruchdarstellung der Zweierkomplementen (2^{-n}). Tabelle 13 zeigt die Darstellung der Zahlen in den verschiedenen Registern. In der Tabelle 13 ist zu erkennen, daß die Register unterschiedlich breit sind. Die beiden Akkumulatorregister der DAU (A, B) sind 56 Bit breit. Somit können 24 Bit Daten multipliziert werden, ohne mit einem Überlauf

arbeiten zu müssen. Die Bits der Akkumulatoren können in zwei Blöcken zu 24 Bit und in einem Block zu 8 Bit einzeln angesprochen werden. Die Inhalte der Akkumulatoren werden für die arithmetischen Operationen der DAU verwendet. Für das Zwischenspeichern von Daten stehen vier 24 Bit breite Datenregister zur Verfügung (X0, X1, Y0, Y1). Diese sind jeweils mit dem X- oder Y- Datenbus verbunden.

Für die Adressierung von Daten stehen vierundzwanzig Register zur Verfügung. Da in jedem Speicherblock 64 kByte Speicher adressiert werden, reichen 16 Bit breite Register zur Adressierung aus. Acht der Register (R0–R7) stehen als Adressregister zur Verfügung. Zu jedem dieser Adressregister gibt es ein Basisadressregister (englisch: offset register). Die Summe des Basisadressregisters und des Adressregisters kann als Adresse benutzt werden. Zusätzlich gibt es zu jedem Adressregister auch noch ein Adress- Modifizierungs- Register (englisch: adress modifier register). Das Modulo des Inhalt eines Adressregisters und des Adress- Modifizierungs- Registers kann als Adresse für einen 'Ringspeicher' oder etwas ähnlichem dienen (diese Ringspeicher werden z.B. bei der schnellen Fouriertransformation (FFT) und anderen Filterprogrammen verwendet).

Die Programmsteuereinheit besitzt sechs weitere Register. Das Statusregister speichert einige arithmetische Eigenschaften der letzten Operation in der DAU (zum Beispiel einen Überlauf, ein Ergebnis von Null,...). Das zweite Register, der Programmzähler (englisch: program counter, PC), enthält einen Zeiger auf die nächste Instruktion, die verarbeitet werden soll. Zwei Register dienen zur Steuerung von Programmschleifen. Das Schleifenzähl- Register enthält die Anzahl noch auszuführender Schleifen und das Schleifen- Adress- Register zeigt mit einem Zeiger auf das Ende der Schleife. Der Anfang der Schleife wird in einem Kellerspeicher (englisch: stack) abgelegt. Der Kellerspeicher besteht aus sechzehn Registern. Jedes dieser Register ist 32 Bit breit. Der Kellerspeicher dient der Ablage von Statusregister, Schleifenregister und Programmzähler bei Unterprogrammen oder geschachtelten Schleifen. Ein Zeiger im Kellerspeicherregister (englisch: stack pointer, SP) zeigt auf die nächste freie Zelle im Kellerspeicher. Dieses Kellerspeicherregister (SP) wird bei der Ablage oder Entnahme von Daten aus dem Kellerspeicher höher bzw. tiefer gesetzt.

Der Befehlssatz besteht aus 62 verschiedenen Instruktionen. Eine Instruktion besteht aus ein oder zwei 24 Bit Worten. Die niederwertigen 8 Bits des ersten Wortes legen in der Regel die Operation fest. Die restlichen Bits dienen der Kodierung der Daten oder Adressen, die in der Instruktion verwendet werden. Es stehen diverse Adressierungsarten für die Daten zur Verfügung.

Durch seinen Aufbau scheint der DSP56001 eher ein Mikrocomputer als ein Mikroprozessor zu sein. Wenn man die externe Speicherschnittstelle nicht benutzt, müssen nur 24 Anschlüsse verdrahtet werden. Zum Betreiben des DSP56001 werden außer einer externen Taktquelle von maximal 20MHz noch eine kleine Reset- Logik und einen Leitungstreiber für die Schnittstelle zu einem übergeordneten Rechner benötigt. Ein komplettes System mit einer ADM Schnittstelle zu einem IBM-AT und 4kByte externer Speicher beschränkt sich auf 26 einfache Logikbausteine.

Leider gibt es keine höhere Programmiersprache für den DSP56001. Es steht jedoch ein leistungsfähiger Makroassembler, ASM56000 von Motorola, zur Verfügung. Für die Untersuchungen dieser Arbeit wurde ein Simulator des Prozessors DSP56001 (SIM56000) verwendet. Dieser Simulator kann wie ein DSP56001 System programmiert werden. Er lief auf einem IBM-AT unter dem Betriebssystem PC-DOS 3.2. Desweiteren stand eine Entwicklungsplatine zur Verfügung, welche über einen Benutzerbus mit einem IBM-AT verbunden war. Diese Platine besaß jedoch nur 4kByte Speicher, was den Einsatz stark einschränkte.

Folgende Eigenschaften des DSP56001 erschweren den Einsatz dieses Prozessors als Trigger:

1. Der Befehlssatz dieses Prozessors ist klein, wodurch die Quellprogramme lang und unübersichtlich werden.

2. Eine Programmierung in einer höheren Programmiersprache ist nicht möglich.

3. Durch die interne Harvardstruktur muß der Programmierer stärker als bei anderen Prozessoren die Möglichkeiten von parallelen Prozessen berücksichtigen. Wenn zum Beispiel für eine Operation ein Bus nicht belastet wird, sollte der Programmierer diesen Bus benutzen für Daten, welche später benötigt werden.

4. Wie bereits oben erwähnt, werden einige Funktionen in eigenständigen Einheiten ohne Belastung der Programmsteuerungseinheit ausgeführt. Einige dieser Funktionen werden nicht in einem Arbeitszyklus ausgeführt. So muß ein Programm Warteschleifen für die Programmsteuereinheit enthalten, um das Funktionsergebnis abzuwarten.

5. Der Befehlssatz enthält eine Menge an typischen DSP Befehlen, welche man von anderen Mikroprozessoren nicht kennt. Dies verlangt eine gewisse Eingewöhnung für einen Assembler- Programmierer. Dies gilt z. B. für Befehle, welche in einem Schritt Addieren und das Ergebnis Rollen oder in einem Schritt Addieren und Multiplizieren.

6. Die Aufteilung des Speichers in drei Blöcke (P-, X- und Y- Speicher) erfordert sorgfältige Planung, wo die Daten abgespeichert werden.

Folgende Eigenschaften begünstigen den Einsatz dieses Prozessors als Trigger:

1. Der Aufwand zum Aufbau eines kompletten Systems ist sehr gering.

2. Für Aufgaben mit vielen arithmetischen Operationen ist der Prozessor sehr schnell.

3. Der Makroassembler ist sehr benutzerfreundlich und leistungsfähig.

4. Das Simulationspaket des DSP56001 auf dem IBM-AT ist ein hervorragendes Mittel zum Testen von Programmen und Routinen.

5. Der Befehlssatz ist zwar eingeschränkt, jedoch sind einige der Instruktionen sehr leistungsfähig.

6. Die Firma Motorola bietet einige leistungsfähige Peripheriebausteine und Programme für den DSP56001 an.

7. Der Aufbau eines DSP56001 Systems ist technisch einfach zu realisieren. Komplett industriell gefertigte Syteme mit einer VME Busschnittstelle sind lieferbar.

I.2. Zukünftige Verbesserungen des DSP56001

Die Firma Motorola hat angekündigt, eine neu überarbeitete Version des DSP56001 im Frühjahr 1988 liefern zu können. Diese Version wird mit hoher Wahrscheinlichkeit eine maximale externe Taktfrequenz von 40MHz besitzen, was eine Verdopplung dieser zu der alten Version darstellt. Durch interne Verbesserungen wurden einige Befehle zeitlich beschleunigt. Insgesamt soll die neue Version viermal mehr elementaren Instruktionen pro Sekunde ausführen (MIPS) als die alte. Es ist leider auch in Zukunft nicht mit einer höheren Programmiersprache für den DSP56001 zu rechnen.

II Der Universalprozessor MC68020

II.1. Beschreibung des MC68020

Der MC68020 ist aufgebaut auf einem registerorientierten Konzept. Dennoch stehen drei Kellerspeicher (englisch: interupt stack, master stack and user stack) im Arbeitsspeicher zur Verfügung. Diese enthalten Daten für die Systemüberwachung und Systemkontrole (englisch: supervisor mode), für Systemunterbrechungen (englisch: interupts) und Unterroutinen.

Die interne Datenbreite beträgt maximal 32 Bit (für die Multiplikation und Division auch 64 Bit (englisch: quad words)). Der Prozessor unterstützt verschiedene Datenformate (BCD (englisch: binary-coded dezimal), 1Bit, 1 Byte, 1 Wort (= 2 Byte), ein Langwort (= 4 Byte) und variable lange Bit- Felder (englisch: bit fields)). Die arithmetische Einheit betreibt Festkommaarithmetik. Zusätzlich besitzt der MC68020 eine Schnittstelle zu einem sekundären Prozessor (Co-Prozessor) für Fließkommaarithmetik. Motorola vertreibt einen speziell angepaßten Co- Prozessor (MC68881), desen Datenformat der IEEE 754 Norm (32-Bit Gleitkommadarstellung) entspricht.

Die Abarbeitung der Instruktionen erfolgt in mehreren Schritten, gleich einem Fließband (englisch: pipelining). Als erstes wird die Instruktion auf das "Fließband" (pipeline) gelegt (englisch: fetch). Als nächstes prüft eine eigenständige Logik, ob die Instruktion schon vollständig ist oder weitere Bytes geladen werden müssen. Diese Vorgänge laufen gleichzeitig mit der Ausführung der vorherigen Instruktion ab und erhöhen somit die Ausführungsgeschwindigkeit. Diese Aufgabe wird von dem Instruction Prefetcher und Decoder eigenständig ausgeführt.

Ein Sequenzer führt die Instruktion aus, in dem eine kleine interne mikrokodierte Prozedure gestartet wird. Der Sequenzer wird gesteuert von der Kontrolleinheit, welche auch die Register enthält. An Registern stehen je 8 Daten- und Adressregister für die 18 verschiedenen Adressierungsarten zur Verfügung. Zur Steuerung des schnellen Instruktionsspeichers (instruction cache) stehen zwei weitere Register zur Verfügung. Drei Kontrollregister, der Instruktionszeiger, das Statusregister und das Zustandsregister (englisch: condition code register), dienen dem geordneten Programmablauf.

Die interne Prozedure, welche der Sequenzer ausführt, kann direkt von der Instruktions- Einheit (englisch: instruction execution unit) dekodiert und ausgeführt werden. Diese Einheit besteht aus der Adressberechnungseinheit (englisch: CAU, control arithmetic unit) und der Datenarithmetikeinheit (englisch: DAU, data arithmetic unit)

Der externe Adress- und Datenbus wird synchronisiert und gesteuert von der Bus Steuerung (englisch: bus controller). Diese regelt die asynchrone Übertragung.

Die letzte eigenständige Einheit des MC68020 ist der schnelle Instruktionsspeicher (instruction cache). Dieser liest den externen Datenbus und speichert Instruktionen zwischen.

Der Umfang der verschieden Instruktionen ist sehr groß. Diese können eingeteilt werden in 7 Gruppen:

- arithmetische und logische Operationen
- Schiebeoperationen für Daten
- Operationen, welche den Programmablauf steuern (z.B. Sprung Operationen)
- Operationen, welche Daten testen (z. B. Datenvergleich)
- Operationen, welche Datenformate konvertieren (z.B. BCD)
- Operationen zur Prozessorsteuerung
- Operationen für den Co- Prozessor

Der komplette Instruktionssatz ist in der Literatur [31] und [39] enthalten.

Die Zahl der höheren Programmiersprachen für MC68020 Systeme ist sehr groß (z.B. C, FORTRAN-77, Echtzeit- FORTRAN (englisch: RTF, real time FORTRAN), Lisp, Prolog, Pascal, BASIC). Auch leistungsfähige Assembler sind vorhanden. Viele dieser Assembler erlauben es, das Quellprogramm in Form von S-Records von dem Entwicklungsrechner auf den Zielrechner zu übertragen. Die Industrie bietet zusätzlich eine große Zahl an Programmen und Programmodulen an. Somit ist die Programmierumgebung auf der MC68020 Umgebungen sehr gut.

Die Vorteile des MC68020 bei der Entwicklung von Triggern sind:

1. Gute Entwicklungsumgebung
2. großer Instruktionssatz mit zum Teil komplexen Operationen
3. universelle Datenformate
4. fertige MC68020 Systeme in großer Zahl käuflich zu erhalten
5. sehr anpassungsfähig an externe Bausteine
6. hohe Verbreitungsgrad der Kenntnis des MC68020 Assemblers
7. Anschluß an den VME Bus technisch nicht aufwendig

Die Nachteile des MC68020 bei der Entwicklung eines Triggers sind:

1. großer technischer Aufwand für den Aufbau eines MC68020 Systems
2. nicht voll optimierte arithmetische Einheit für Daten
3. Adressierungsart für kleine Tabellen langsam
4. MC68020 Systeme haben einen hohen Platzbedarf

II.2. Zukünftige Verbesserungen des MC68020

Die Weiterentwicklung des MC68020, der MC68030, bietet für den Einsatz als Trigger keine bedeutende Verbesserung. Es wurde beim MC68030 ein schneller Datenspeicher von 256 Byte (englisch: data cache) auf dem Prozessor eingerichtet, der jedoch keine Verwendung finden kann bei den betrachteten Triggerprogrammen. Eine kleine Verbesserung stellt die auf dem MC68030 integrierte Speicherverwaltung (englisch: memory manegment unit, MMU) dar, weil somit der Platzbedarf des kompletten Prozessorsystems verringert werden kann. Die Taktrate des MC68020 wird sich wohl in naher Zukunft nicht viel verbessern, da diese bereits heute bei 25–30 MHz liegt.

III Des Transputer IMS T414

III.1. Beschreibung des IMS T414

Die Familie der Transputer umfaßt vier verschiedene Mitglieder. Diese unterscheiden sich durch die Datenbreite, dem internen Schreib- Lese Speicher und dem Datenformat. Für die untersuchte Aufgabe ist der Transputer IMS T414 ausreichend.

Der IMS T414 hat eine Datenbreite von 32 Bit. Die Arithmetikeinheit arbeitet in Festkomma- Darstellung. Das Konzept dieses Prozessors ist eine Mischung aus register- und stapelspeicher-orientierter Maschine. Da der interne Schreib- Lese Speicher von 2kByte genau so schnell gelesen und beschrieben werden kann wie ein Register, wird dieser über einen Zeiger als Register verwendet. Dieses Konzept wird Workspace Konzept genannt. Es hat den Vorteil, daß bei dem Wechsel des aktuellen Programmes bei quasi parallel ablaufenden Programmen (englisch: time sharing) nur der Workspace- Zeiger geändert werden muß, um die jeweilige Umgebung des neuen aktuellen Programmes herzustellen. Der Workspace- Zeiger ist im Workspace-Register abgelegt. Der Schreib- Lese- Speicher ist mit einem schnellen internen Daten- und Adressbus mit der zentralen Steuereinheit verbunden.

Die zentrale Steuereinheit besteht aus zwei Untereinheiten. Die erste Untereinheit, die Systemkontrolle (siehe Abbildung 45), erzeugt aus dem externen Takt von 5 MHz den internen Takt von 20 MHz. Hierzu wird eine Schaltung verwendet, die nur noch einen prozessor-externen Kondensator benötigt. Solche Schaltungen werden PLL- Schaltung (englisch: phase locked loop) genannt. Als zweites wertet die Systemkontrolle die externen Anschlüsse für Fehlerfälle aus.

Die zweite Untereinheit ist der zentrale Prozessor. Der Prozessor enthält die Register und die zentrale Recheneinheit. Es existieren lediglich 7 Register. Die Bedeutung des Workspaceregisters wurde bereits erwähnt. Das zweite Register ist der Befehlszeiger (englisch: instruction pointer register). Dieses Register enthält einen Zeiger auf die Speicherstelle mit der nächsten auszuführenden Instruktion. Das dritte Register, das Operandenregister, wird benötigt, um Operanden und indirekte Instruktion aufzubauen. Die restlichen Register, A, B und C, bilden einen dreielementigen Kellerspeicher (englisch: LIFO, last in first out ‚push down store). Kellerspeicher Lade-Befehle (push) verschieben den Inhalt von Register B nach C und A nach B, bevor der neue Wert in A abgelegt wird. Dagegen hebt ein Lesebefehl (pop), nachdem der Wert von A ausgegeben wurde, den Inhalt des Registers B in das Register A und stapelt den Inhalt des Registers C nach B hoch. Arithmetische und logische Operationen arbeiten ausschließlich auf den dreielementigen Registerstapel. Zum Beispiel addiert die ADD- Instruktion die beiden obersten Elemente, der Inhalt des A und B Registers, und legt das Ergebnis im A Register ab. Die Operation "verbraucht" also zwei Register und "füllt" eins wieder auf. Somit muß der Inhalt des Registers C noch in das Register B angehoben werden. Die Inhalte solcher "verbrauchten" Register sind undefiniert und dürfen vom Assemblerprogrammierer nicht nocheinmal verwendet werden.

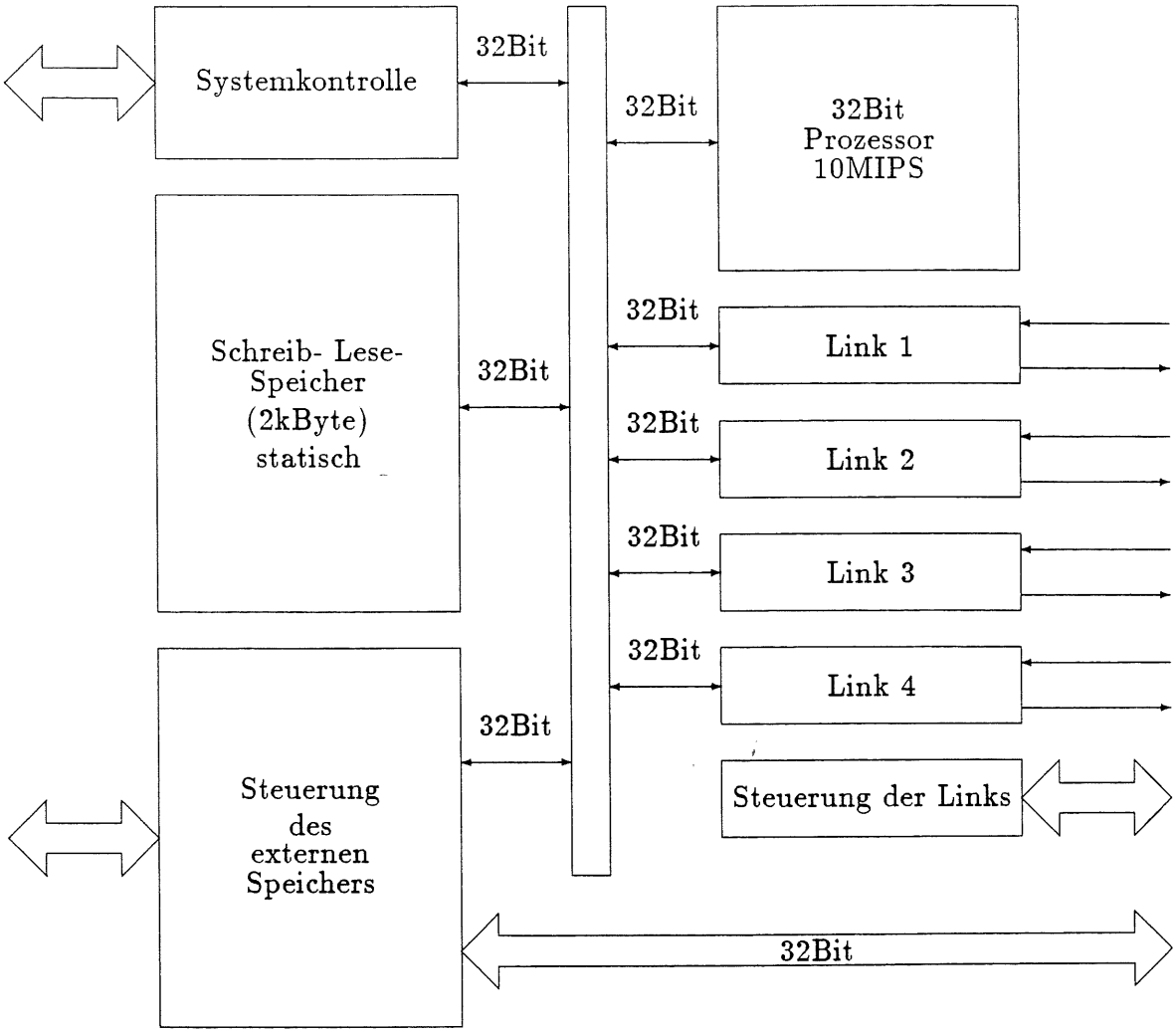


Abbildung 45: Der interne Aufbau des T414 (Bockschaltbid)

Die zentrale Recheneinheit wird sowohl für die Adressberechnung als auch für die mathematischen und logischen Operationen verwendet (ist also sowohl DAU als auch CAU[10]). Mit einer Adressbreite von 32 Bit kann die Recheneinheit vier Gigabyte (2^{32}) Speicher linear adressieren. Der interne Speicher gehört hierzu und liegt am Anfang des Adressbereiches.

Über den internen Daten- und Adressbus ist eine Speicherverwaltung (englisch: EMI, external memory interface) mit der Recheneinheit (CAU/DAU) verbunden. Über diese EMI kann der Speicher extern auf die vier Gigabyte erweitert werden. Zugriffe auf den externen Speicher sind, bedingt durch die höheren kapazitiven Leitungslasten und der zweifachen Verwendung externer Prozessoranschlüsse für sowohl den Daten- wie auch dem Adressbus (englisch: data and adress multiplexing), sehr viel langsamer als auf dem internen Speicher. Die EMI liefert desweiteren die Signale, um eventuell vorhandene dynamische Schreib- Lese- Speicher wieder aufzufrischen. Da keine spezielle Adressberechnungseinheit existiert, erfolgt die Adressierung des Speichers nach den Regeln der vorzeichenbehafteten Festkomma- Darstellung. Der interne Speicher belegt also die "negativsten" Adressen von $80000000_{Basis\ 16}$ bis $800007FF_{Basis\ 16}$. Der Rest des Adressraumes wird vom externen Speicher belegt. Die Adresse Null liegt folglich in der Mitte des Adressraumes bei zwei Gigabyte. Die ersten 48 Adressen des Adressraumes sind belegt durch prozessorinterne Adressen, wie die Kanaladressen der vier Links (neun Kanäle = vier Eingangskanäle + vier Ausgangskanäle + ein Kanal für Unterbrechungen (englisch: interrupts)), zwei Zeiger für Zeitgeberschlangen des Verteilmechanismusses (englisch: timer queues) sowie die Register.

Der Speicher ist grundsätzlich in Bytes organisiert. Die nächstgrößere Einheit ist ein 32 Bit Langwort. Die Wortgrenzen müssen immer auf einer durch vier teilbaren Adresse liegen. Der Transputer besitzt einen normalen "bytesex". Das heißt, er ordnet die vier Bytes in der Reihenfolge ihrer Wertigkeit und in Übereinstimmung mit der aufsteigenden Adressierung an. Ebenso erfolgt die Zählung der Bits in einem Byte und im Wort in Übereinstimmung mit ihrem zunehmenden numerischen Gewicht. Bei Lese- Operationen werden stets vier Byte (32Bit) gelesen. Schreiboperationen sind Byte und vier Byte weise möglich.

Als letzte Einheit sind die Links mit dem internen Daten- und Adressbus verbunden. Diese Links sind eine eigenständige Entwicklung der Firma Inmos. Sie dienen der Kommunikation mit anderen Prozessoren oder Rechnersystemen. Eine Kommunikation ist in beiden Richtungen möglich (bidirektional). Dabei werden die 32Bit nicht zeitlich gleichzeitig (parallel), sondern zeitlich hintereinander (seriell) übertragen. Die Geschwindigkeit der Übertragung ist von 5 Megabit pro Sekunde (= MBAUD)(dies entspricht 160000 Langwörter pro Sekunde) bis 20 Megabit pro Sekunde (entspricht 640000 Langworte pro Sekunde) wählbar. Diese Links sind eigenständige Prozessorbausteine, welche Daten übertragen, ohne den internen Bus zu belasten. Somit kann der T414 gleichzeitig Daten auf dem internen Bus verschieben,

[10] siehe dazu auch die Beschreibung des DSP56001 in Kapitel 5.1.

wärend Daten von vier anderen Prozessoren ein- oder auslaufen.

Da die Links von Inmos ein nicht genormtes Übertragungsprotokoll besitzen, sind von Inmos verschiedene Link Adapter entwickelt worden. Diese konvertieren das Protokol eines 8 Bit Parallelen Busses in das eines Transputer- Links (IMS C001, IMS C002, IMS C011, IMS C012). So ist es möglich, die Links als Bus zu einem übergeordneten Rechner (englisch: host interface) zu verwenden.

Der Befehlssatz des IMS T414 ist ungeeignet für das manuelle Programmieren in Assembler und ist optimiert zur einfachen Übersetzung von Hochsprachenprogrammen (z.B. durch Compiler). Alle 103 Befehle des T414 besitzen das gleiche Format. Jede Instruktion besteht aus einem einzelnen Byte. Von diesem Byte sind vier Bit für die Befehlsdekodierung und vier Bit für die Daten reserviert. Mit vier Bit können nur 16 verschiedene Befehle oder Datenwerte linear kodiert werden. Dreizehn dieser sechzehn möglichen Befehlskode stellen die am häufigsten benötigten Instruktionen zur Verfügung, die man auch direkte Funktionen nennt. Zu den wichtigsten Operationen in einem Programm gehören das Laden oder Speichern von Variablen und Konstanten sowie verschiedene Sprünge. Der Befehl *load constant* zum Beispiel erlaubt es, Werte von 0 bis 16 mit nur einer ein Byte großen Instruktion auch in 32 Bit breite Speicherzellen zu laden.

Die Befehle *load local* und *store local* adressieren relativ zum Workspacezeiger, wobei für ein Zugriff auf die ersten 16 Wörter ebenfalls nur ein einziges Instruktionsbyte notwendig ist. Die Operationen *load non local* und *store non local* arbeiten ähnlich, nur daß sie den Inhalt von Register A als Basiszeiger verwenden.

Zwei der direkten Funktionen zeichnen sich dadurch aus, daß sie als Präfixfunktionen nur das Operandenregister modifizieren. Die Instruktionen *prefix* und *negativ prefix* können dort beliebig große Operanden zusammensetzen. Das 32 Bit breite Operandenregister ist Anfangs völlig leer, denn Präfixbefehle sind die einzigen Befehle, die dieses Register nicht nach der Befehlsausführung löschen.

Der Befehl *prefix* lädt seine vier Datenbits (englisch: "nibble") in das Operandenregister und verschiebt den Inhalt des Operandenregisters auch gleich vier Bits nach links, so daß die vier niederwertigsten Bits sofort wieder frei sind. Der Befehl *negativ prefix* funktioniert ebenso, nur wird das Operandenregister vor dem Verschieben noch komplementiert ($1 \rightarrow 0$, $0 \rightarrow 1$). Alle direkten Funktionen des Transputers laden nun bei der Ausführung ihre vier Datenbits in die niederwertigsten vier Bits des Operandenregisters, welches dann als Operand verwendet wird. Somit können auch Operanden von mehr als vier Bits aufgebaut und verarbeitet werden. Der aufgebaute Operand wird jedoch nach der Funktion sofort wieder gelöscht.

Dem letzten Befehl aus den 16 direkten Befehlen kommt eine besondere Bedeutung zu. Der Befehl *operate* öffnet das Tor zu den weiteren Instruktionen (87) des Transputers. Auch die vier Datenbits dieses 8 Bit Befehles werden in das Operandenregister geladen. Das Operanden Register wird jedoch nicht als Operand ausgewertet, sondern wird als auszuführende Operation interpretiert.

Auf diese indirekte Weise sind weitere sechzehn Instruktionen mit nur einem Byte Kode zugänglich. Diese Instruktionen beinhalten die häufigsten Funktionen, welche

als Operanden die Werte im Registerstapel verwenden. Darunter ist auch der *sub* Befehl, mit dem der Inhalt des B- Registers von Inhalt des A- Registers subtrahiert wird. Das Ergebnis dieser Funktion wird in das A- Register geschrieben. Der Inhalt des C- Registers wird in das B- Register geschrieben.

Die Programmierung des IMS T414 muß jedoch nicht in Assembler erfolgen. Als höhere Programmiersprache stehen C, FORTRAN-77, Pascal und Occam zur Verfügung. Bislang besitzt jedoch nur die Sprache Occam die Möglichkeit, ein Programm parallel abarbeiten zu lassen. Jedoch besteht die Möglichkeit der Kombination der verschiedenen Sprachen. So kann ein Programm ein Occam- Gerüst besitzen, welches festlegt welche Routinen parallel oder seriell verarbeitet werden können, die Prozeduren jedoch sind in C, FORTRAN-77 oder Pascal geschrieben. Die Entwicklung der Programme kann erfolgen in Occam, FORTRAN-77, Pascal oder C auf folgenden Entwicklungsrechnern (englisch: host):

- Stride 440/450 (68000 System)
- VAX/VMS 11/7xx eingeschlossen der MicroVAX(DEC- System)
- IBM PC, XT und AT (IBM Personel Computer System)
- Entwicklungsplatinen, welche über serielle Schnittstellen (V24/RS232) mit Entwicklungsrechner verbunden werden.

Der Aufbau eines Transputersystemes ist recht einfach, da die EMI, die zentrale Steuereinheit und die Links schon alle benötigten Signale bereitstellen. Die geringe Leistungsaufnahme von 0.9 Watt bei 5 Volt Versorgungsspannung im aktiven Zustand bereitet keine Probleme bei der Kühlung. Folgende Eigenschaften der Transputer erschweren den Einsatz solcher als Trigger:

1. Eine Programmierung in Assembler ist nahezu unmöglich, bedingt durch das Arbeiten mit Datennibbles.
2. Das Arbeiten in einer Hochsprache ist zur Zeit nur in OCCAM, einer von INMOS vertriebenen Hochsprache für das Programmieren von parallelen Prozessen, effizient. Diese Programmiersprache ist unter Physikern noch nahezu unbekannt.
3. Durch den stark reduzierten Instruktionssatzes werden die lauffähige Programme sehr lang und unübersichtlich.

Die Vorteile dieses Prozessors beim Einsatz als Trigger sind:

1. Jede Funktion wird in einem internen Taktzyklus ausgeführt. Insbesondere tritt kein 'pipelining' bei Register–Ladebefehlen auf.
2. Die schnellen Links würden den Aufbau eines Mehrprozessorsystems stark vereinfachen. Als Vorteil stellt sich dann die kleine externe Taktfrequenz heraus.

3. Die Firma INMOS hat ein gutes Entwicklungssystem entwickelt, welches über eine der Links des Transputers mit einem IBM-PC/XT/AT kommuniziert. Ein ähnliches Sytem ist nun auch preisgünstig von der Firma HEISE Hannover, HEMA und PARSYTEC entwickelt worden.

4. Mit Occam steht eine schnelle Hochsprache zur Verfügung, in welche auch Prozeduren anderer Hochsprachen eingefügt werden können (FORTRAN, Pascal, C).

5. Ein Transputersystem besitzt einen recht einfachen Hardwareaufbau. Außer einer Spannungsversorgung, eines Taktgenerators, eines externen Speichers und einiger Treiberbausteine ist alles Notwendige bereits auf dem Chip des Prozessors.

6. Die Leistungsaufnahme eines Transputersystemes ist sehr gering.

Leider stand kein Entwicklungssystem für diese Untersuchungen zur Verfügung. Die in dieser Arbeit aufgeführten Ausführungszeiten sind bestimmt worden aus den in der Literatur [24], [26] genannten Ausführungszeiten der elementaren Operationen. Da diese Zeiten jedoch stark von der Operandenlänge abhängen, und dies oft keine Berücksichtigung fand, sind die Ausführungszeiten mit einem großen Fehler behaftet.

III.2. Weiterentwicklungen der Transputer

Eine technische bedeutende Weiterentwicklung der Transputer ist nicht in Sicht. Sie könnte stattfinden durch eine Steigerung der internen Taktfrequenz und vielleicht auch durch Vergrößern der Befehlslänge (englisch: op-code length). Diese Entwicklungen könnten eine erhebliche Leistungssteigerung der Transputer zur Folge haben.

Auf dem Sektor der Softwareentwicklung will INMOS bald ein FORTRAN77 anbieten, welches erweitert ist durch Befehle und Elemente der parallelen Datenverarbeitung. Dies würde sicher die Programmierung stark vereinfachen und die Länge des Quellkodes verkürzen und somit die Übersichtlichkeit vergrößern. Es bleibt abzuwarten, ob ein solches FORTRAN Programm auch schnell genug ist, um Aufgaben der Triggerung zu erfüllen.

IV Die Simulation

IV.1. Ein Beispiel

Dieses Kapitel stellt den Ablauf einer Simulation dar. Es werden die Schritte aufgeführt, die für eine vollständige Simulation des Triggerprogramms notwendig sind. Es wird hierbei angenommen, daß die Monte–Carlo Daten bereits formatiert und auf den COPAM Rechner in der Datei \KERMIT\ASCII.LU1 übertragen worden sind. Hier die Anweisungen auf dem COPAM- Rechner:

C:\ > CD \DSP56000\RUN\LOG

C:\DSP56000\RUN\LOG>\DSP56000\MONIKA\COM-FILE\EVENT4A LU1

C:\DSP56000\RUN\LOG> DIR *.LOG>DIRLU1

C:\DSP56000\RUN\LOG> cd ..\..

C:\DSP56000> \DSP56000\MONIKA\COM-FILE\COMAND.COM \DSP56000\RUN\LOG\DIRLU \DSP56000\RUN.CMD \DSP56000\RUN\LOG\

C:\DSP56000> WVW.BAT

Die Stapeldatei WVW.BAT besteht aus folgenden Befehlen:

```
cd \
echo off
prompt $g
cls
echo Diese Datei startet die Simulations von
echo Monika4a
echo auf dem DSP56000 Simulator
echo .
echo Bitte Verzeichnis  DSP56000\RUN\LOG löschen
echo DESYNET verbinden und rücksetzen
echo on
copy \autoexec.bat \auto.bat
copy \dsp56000\monika\utility\wvw.wvw \autoexec.bat
boot
```

Die Stapeldatei WVW.WVW, welche nach dem Neustart eingelesen wird, enthält folgende Befehle:

```
\system\mode mono \dsp56000\run.bat
```

Die Stapeldatei RUN.BAT, führt folgende Befehle aus:

```
cd \dsp56000
sim56000 \dsp56000\run.cmd
echo off
cls
```

```
\dsp56000\monika\com-file\mergen \dsp56000\run\dir\dirlu1 \dsp56000\run\merged\lu1.sum \dsp5600\run\log\
\dsp56000\monika\com-file\auswert \dsp56000\run\merged\lu1.sum \dsp56000\run\auswert\lu1.asw
\dsp56000\monika\com-file\simtrack \dsp56000\run\merged\lu1.sum \dsp56000\run\auswert\lu1.trk
PROMPT $p$g
cd \
del \autoexec.bat
copy \auto.bat \autoexec.bat
del \auto.bat
cd \kermit
kermit take \dsp56000\run\dir\daten.cmd
\batches\boot.bat
```

Diese Stapeldatei startet die Simulation und wertet die Simulationsergebnisse aus. Danach werden die Daten mit Kermit und einem speziellen Makro (daten.cmd) auf die IBM 3084Q übertragen.
Die Stapeldatei RUN.CMD hat folgende Gestalt:

```
LOAD S monika4.SIM
LOAD \DSP56000\LOOKUP4A.LOD
CHANGE CNT1 0
CHANGE CNT2 0
LOG S \dsp56000\run\log\#4LU1N01.LOG
LOAD \dsp56000\run\daten\#4LU1N01.LOD
GO $40
LOG OFF
LOAD \dsp56000\run\daten\#4LU1N01.LOE
.
.
.
QUIT
```

Das Makro für KERMIT stellt die Verbindung zur IBM 3084Q über den Knotenrechner MICOM her und startet auf der IBM das Kermit. Danach überträgt es die Dateien und beendet die Verbindung.
Somit wäre die Arbeit auf dem IBM- AT Kompatiblen erledigt. Die nachfolgende Befehle dienen der weiteren Auswertung der Simulationsdaten auf der IBM3084Q.

```
SUBH F36WEN.MONIKA.S(VERTEX)
SUBH F36WEN.MONIKA.S(HISTOGR)
SUBH F36WEN.MONIKA.S(CUTS)
SUBH F36WEN.MONIKA.S(REDUCT)
```

Jedoch ist zu beachten, daß die Namen der verwendeten GEP- Dateien und der eingelesenen Ergebnisdateien vorher angepaßt werden müssen. Zudem benötigt das Programm CUTS eine Datei, in der manuell alle rekonstruierten und nicht rekon-

struierten Spuren eingetragen werden. Dies ist notwendig, um die Schwellen für den minimalen Impuls und größten Abstand rekonstruierter Spuren zu bestimmen. Eine Spur galt bei meiner Betrachtung als rekonstruiert, falls mindestens 60% der verwendeten Signale, zu der rekonstruierten Spur gehörten. Die Ergebnisse der Simulation können nun interaktiv mit dem Programm GEP dargestellt werden.

V Programme für den DSP56001

V.1. Flußdiagramm von Assemblerprogramm

Da der Assemblercode des DSP56001 weithin unbekannt ist, ist das verwendete Programm nachfolgend ausführlich als Flußdiagramm dargestellt. Dennoch ist es notwendig bei einzelnen Detailfragen, das ebenfalls der Arbeit angefügte Assemblerprogramm zu untersuchen.

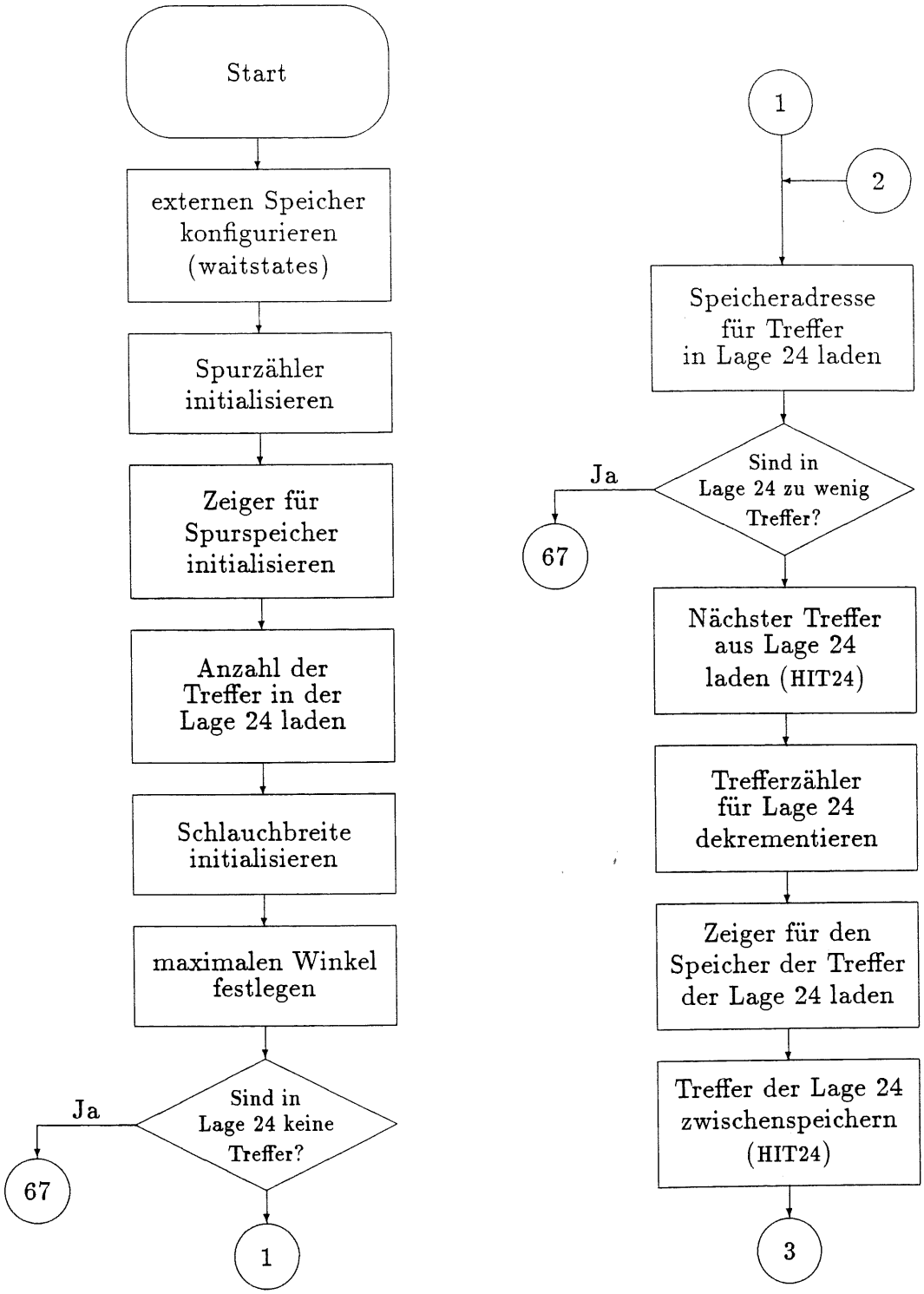


Abbildung 46: Flußdiagramm des verwendeten Programmes (Teil 1)

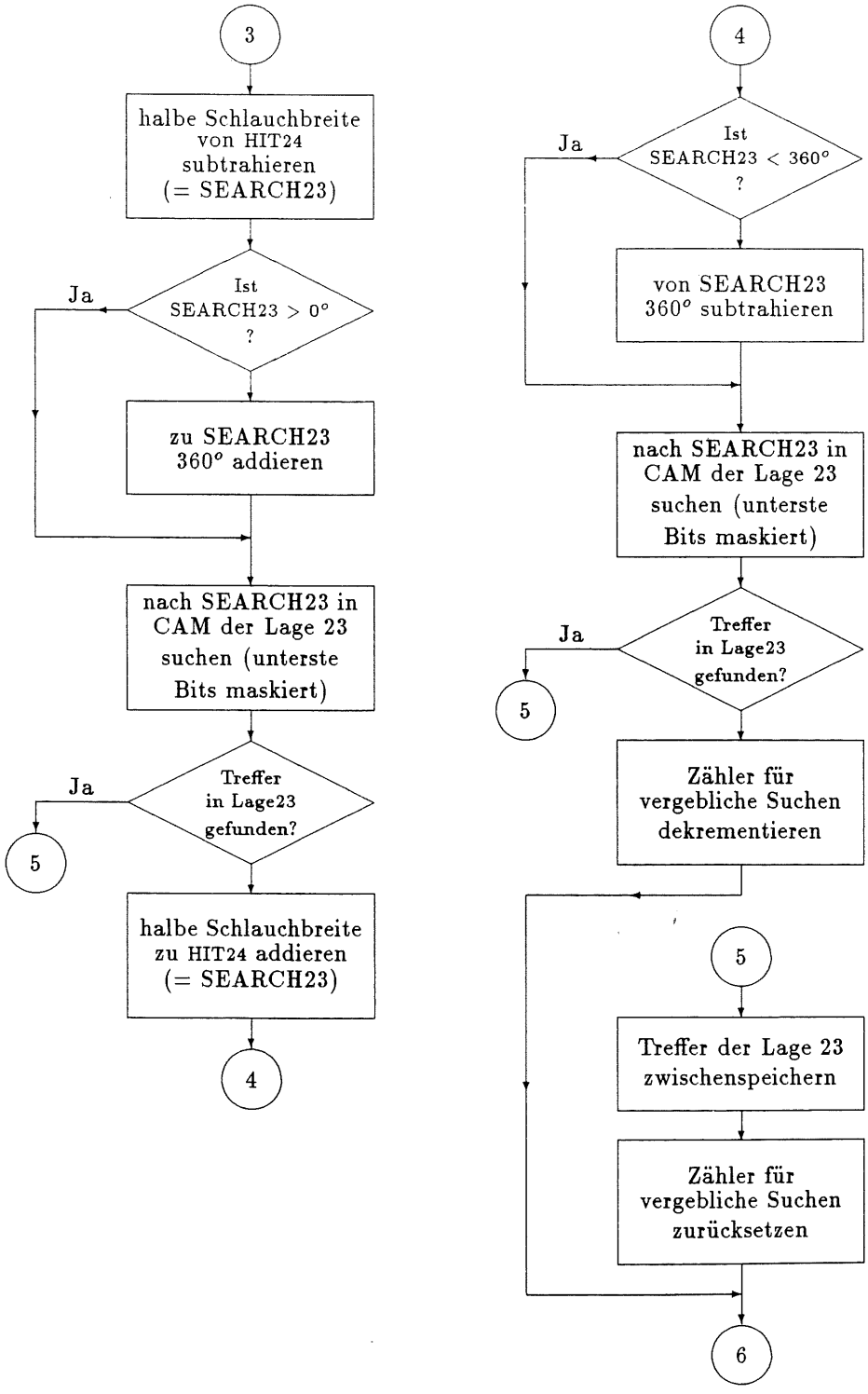


Abbildung 47: Flußdiagramm des verwendeten Programmes (Teil 2)

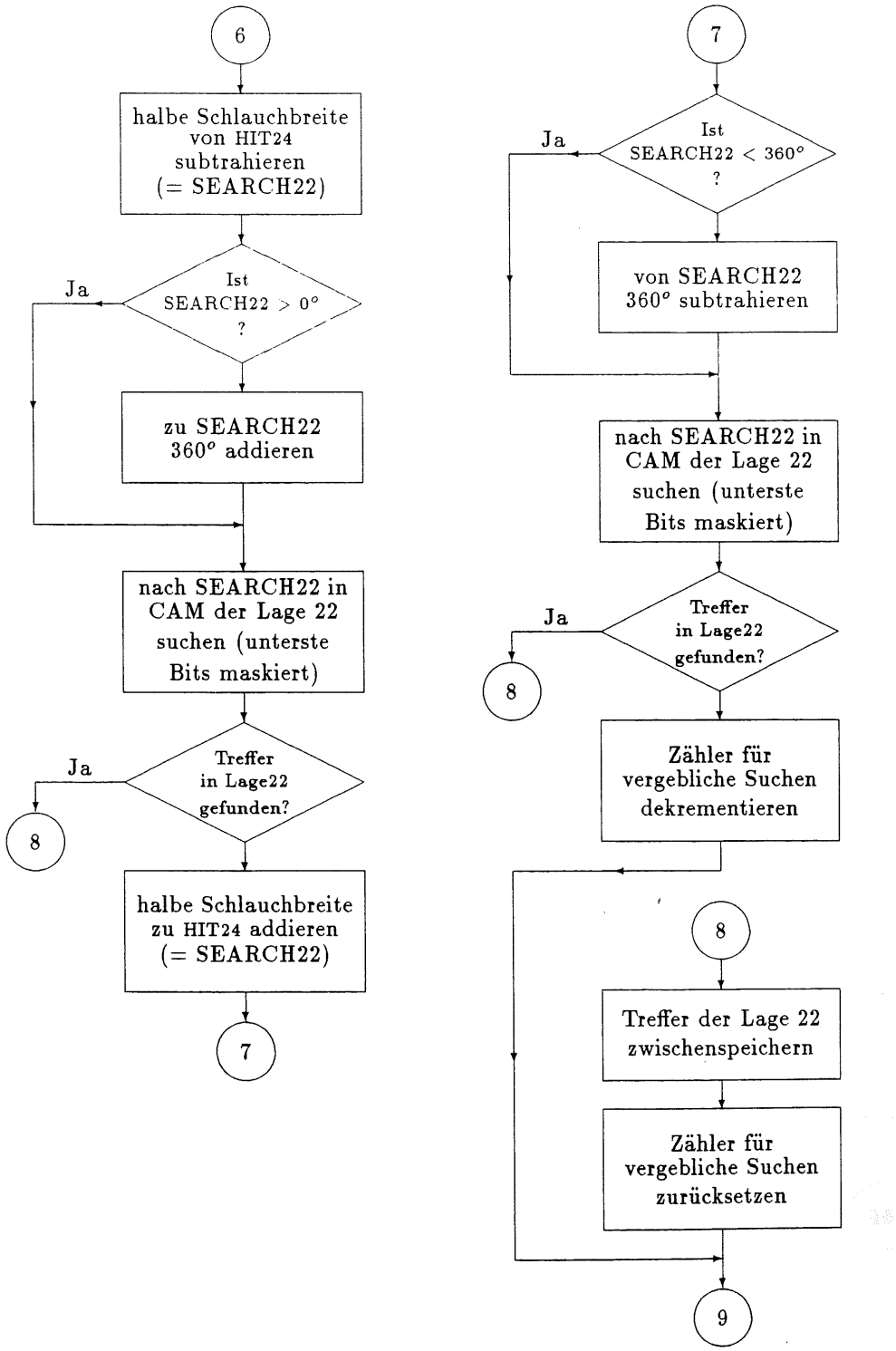


Abbildung 48: Flußdiagramm des verwendeten Programmes (Teil 3)

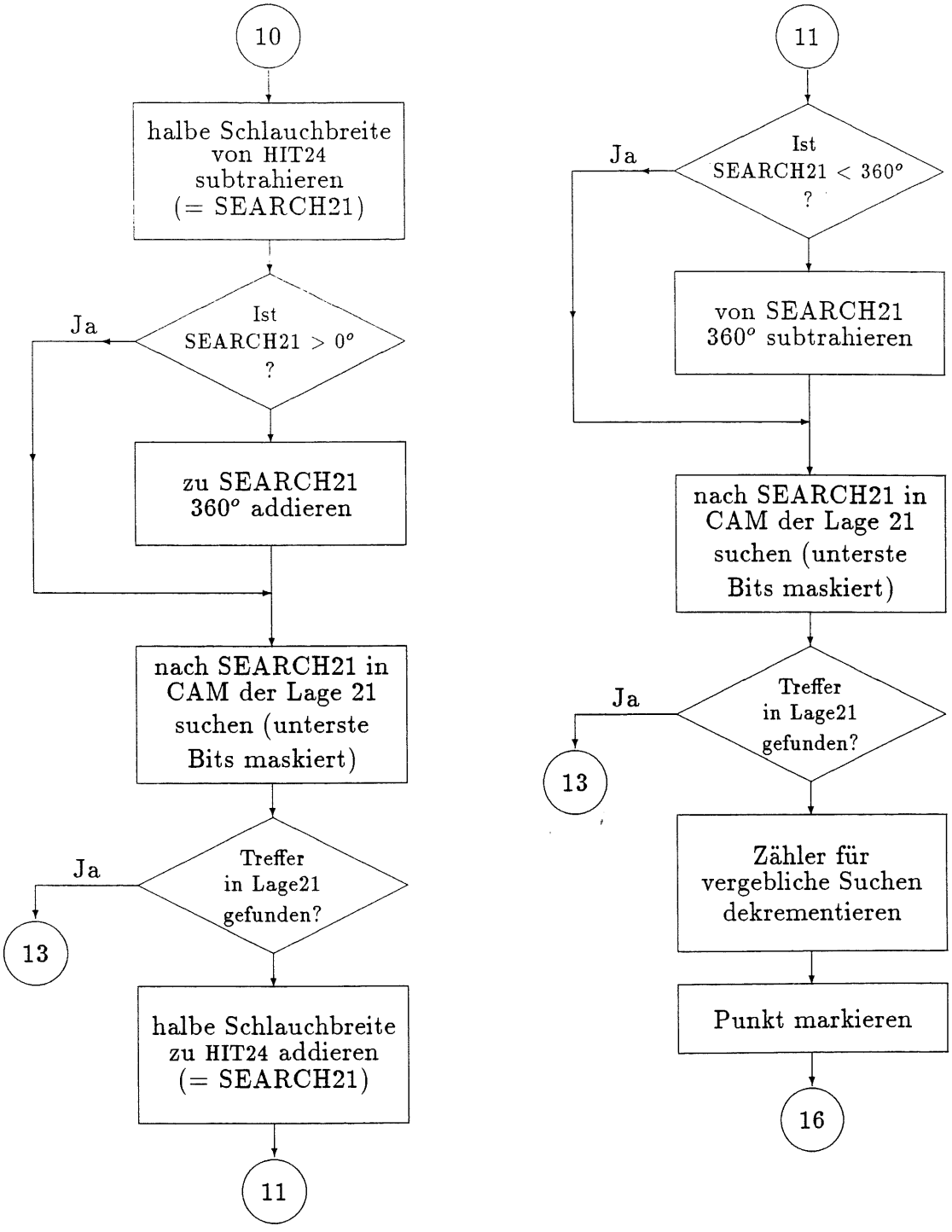


Abbildung 49: Flußdiagramm des verwendeten Programmes (Teil 4)

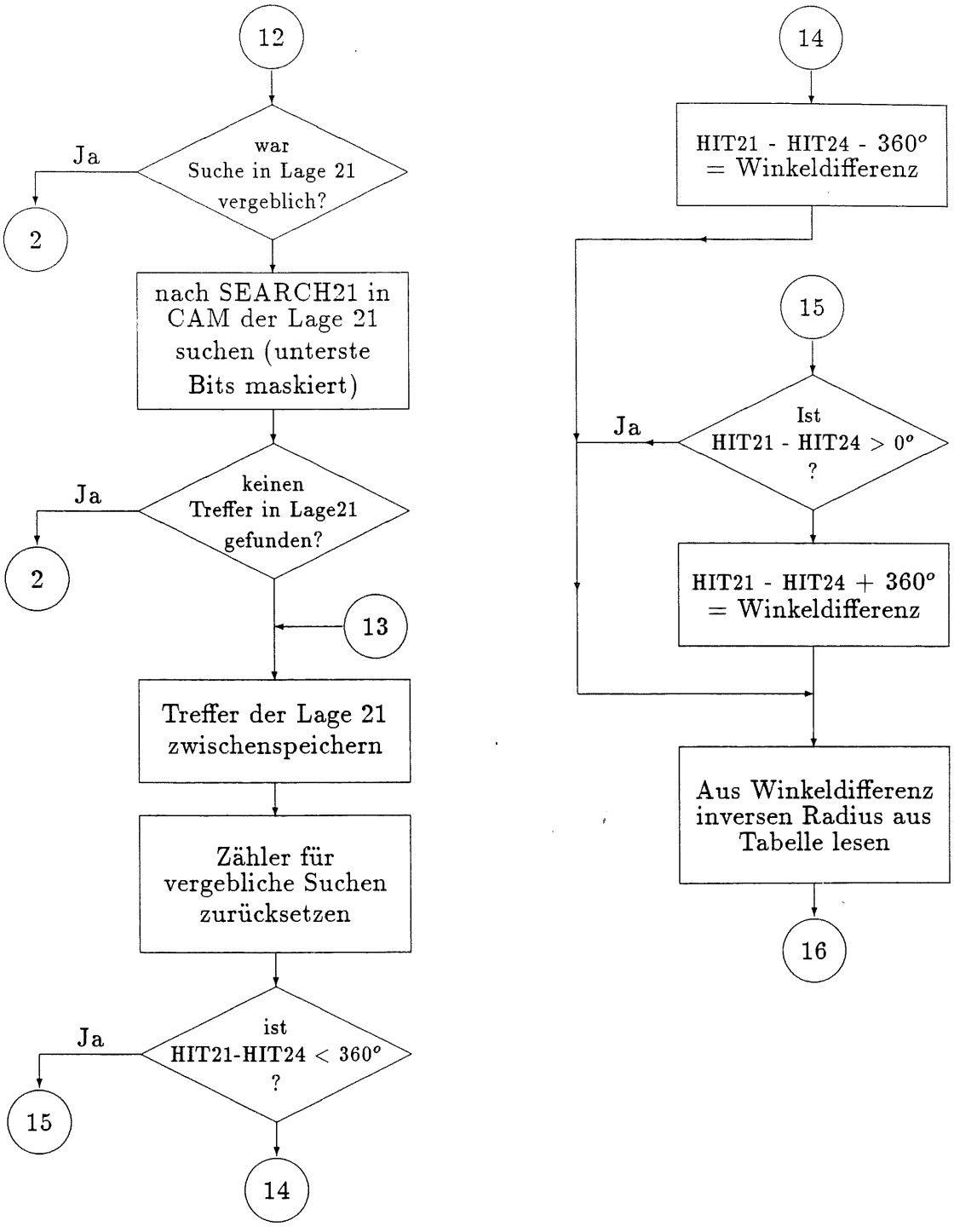


Abbildung 50: Flußdiagramm des verwendeten Programmes (Teil 5)

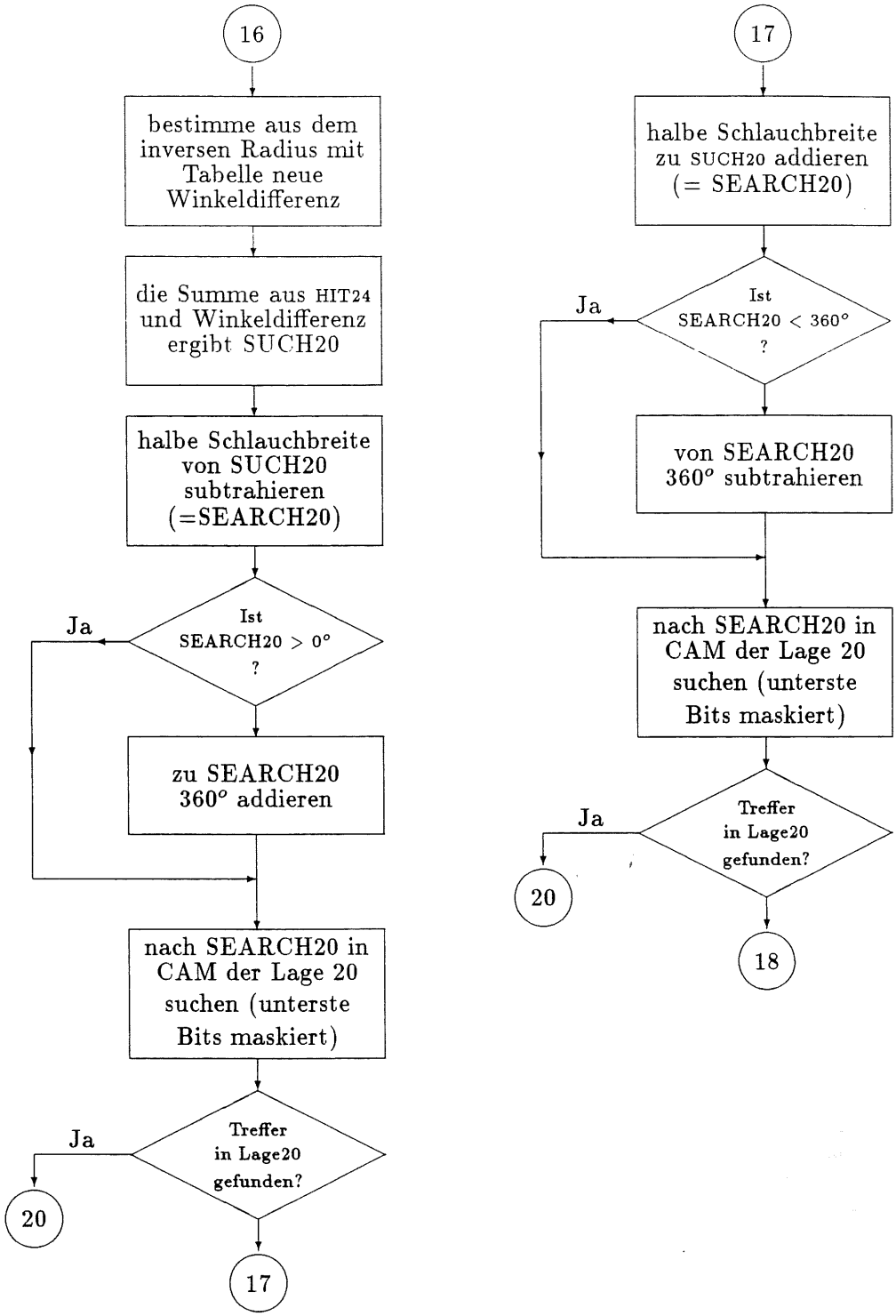


Abbildung 51: Flußdiagramm des verwendeten Programmes (Teil 6)

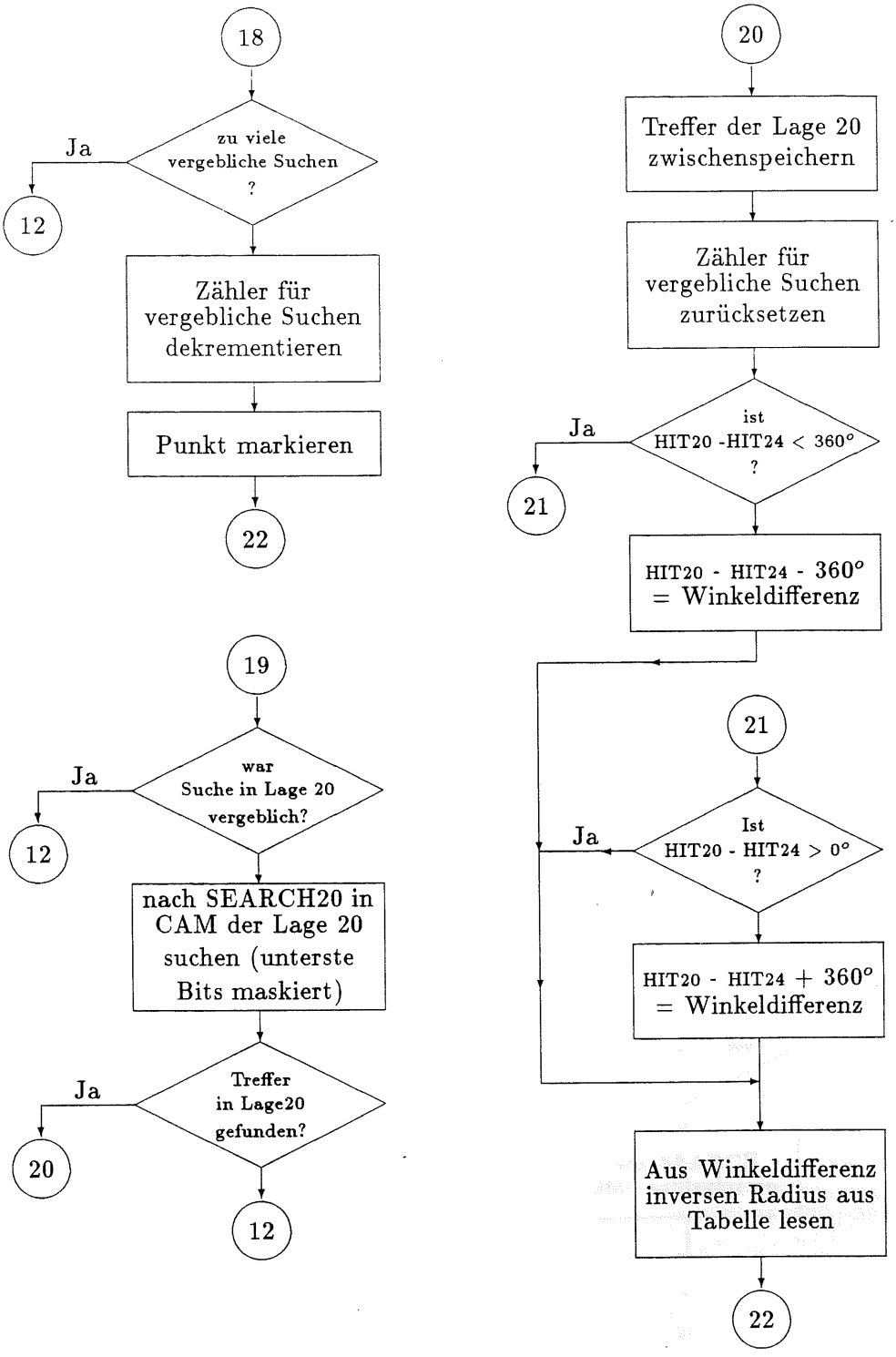


Abbildung 52: Flußdiagramm des verwendeten Programmes (Teil 7)

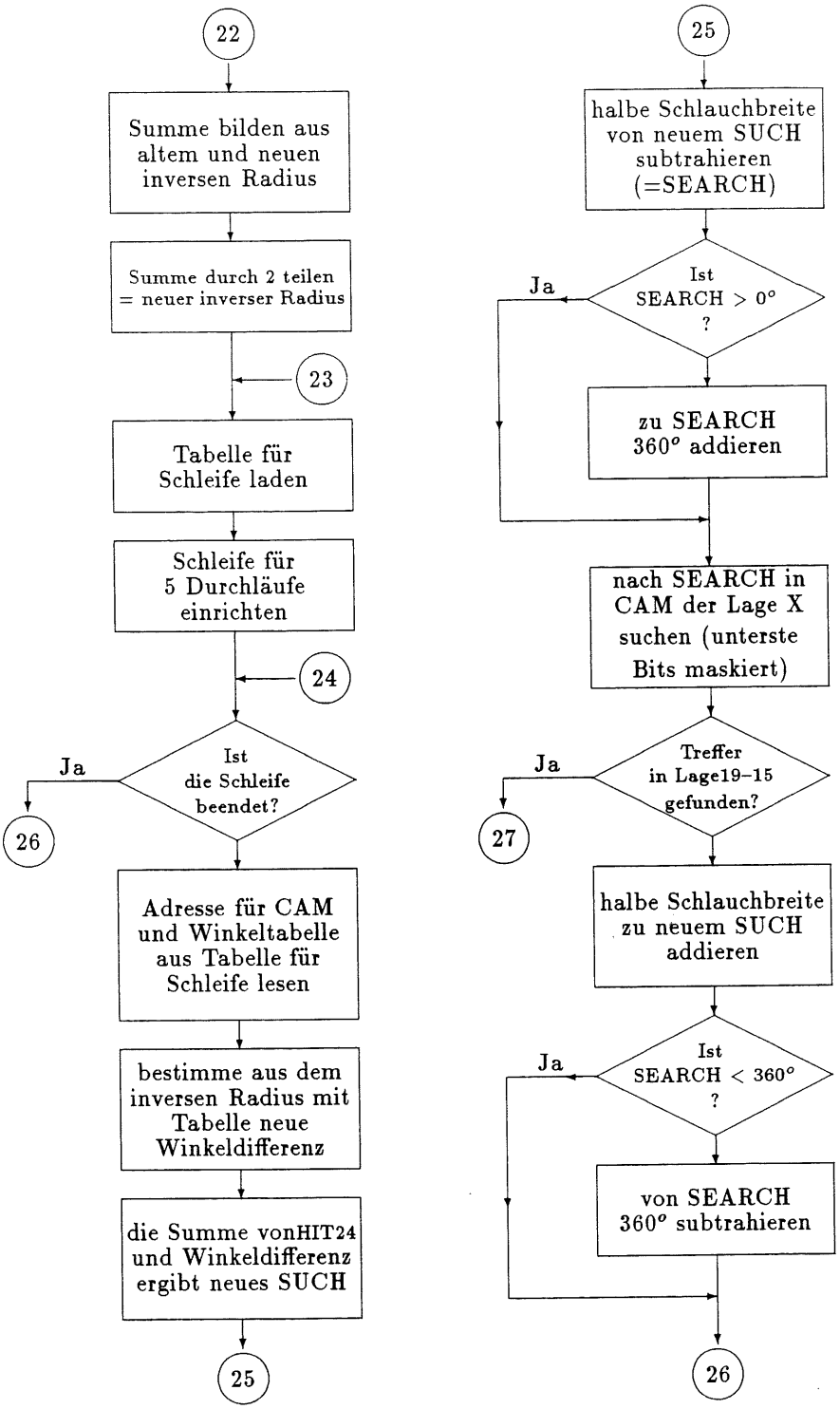


Abbildung 53: Flußdiagramm des verwendeten Programmes (Teil 8)

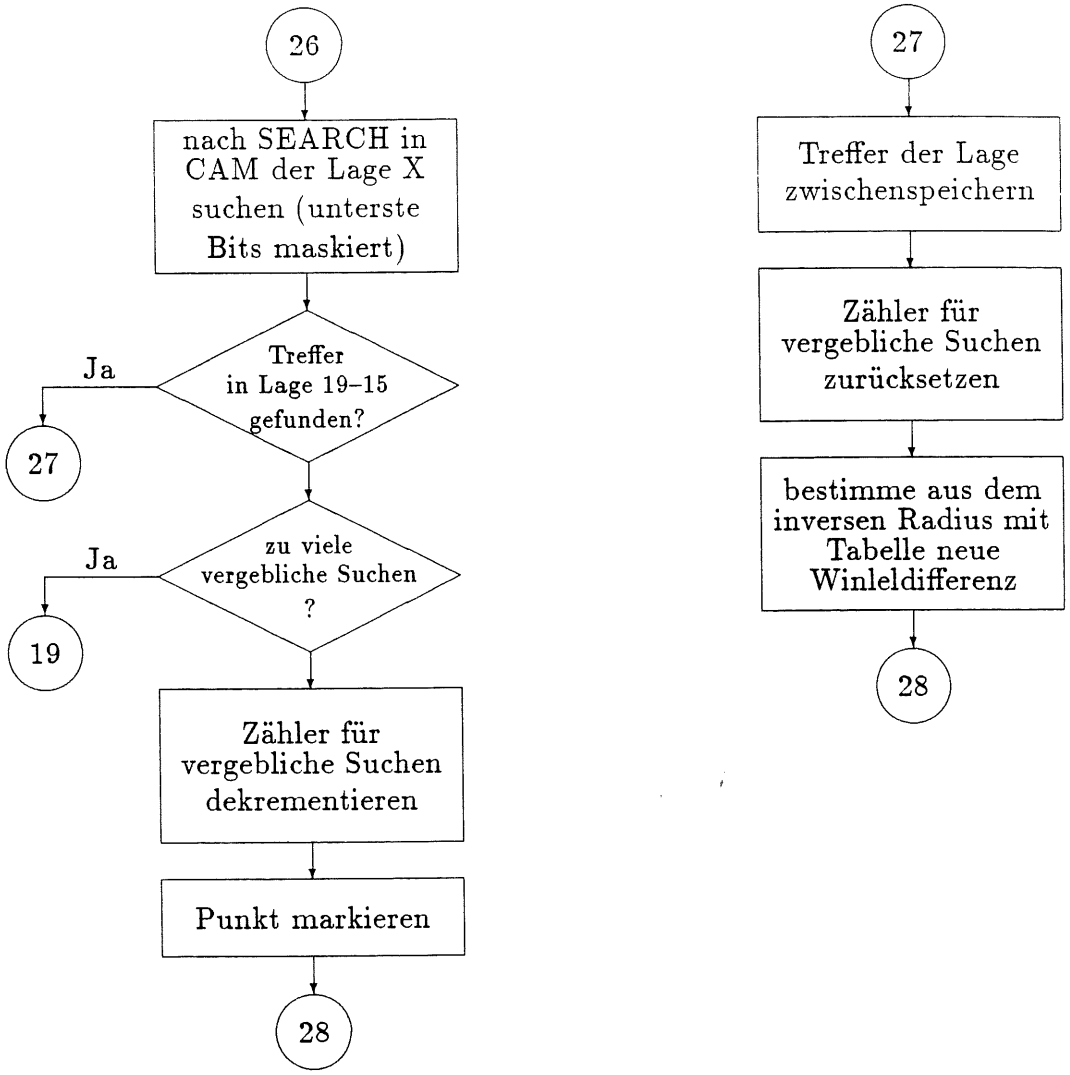


Abbildung 54: Flußdiagramm des verwendeten Programmes (Teil 9)

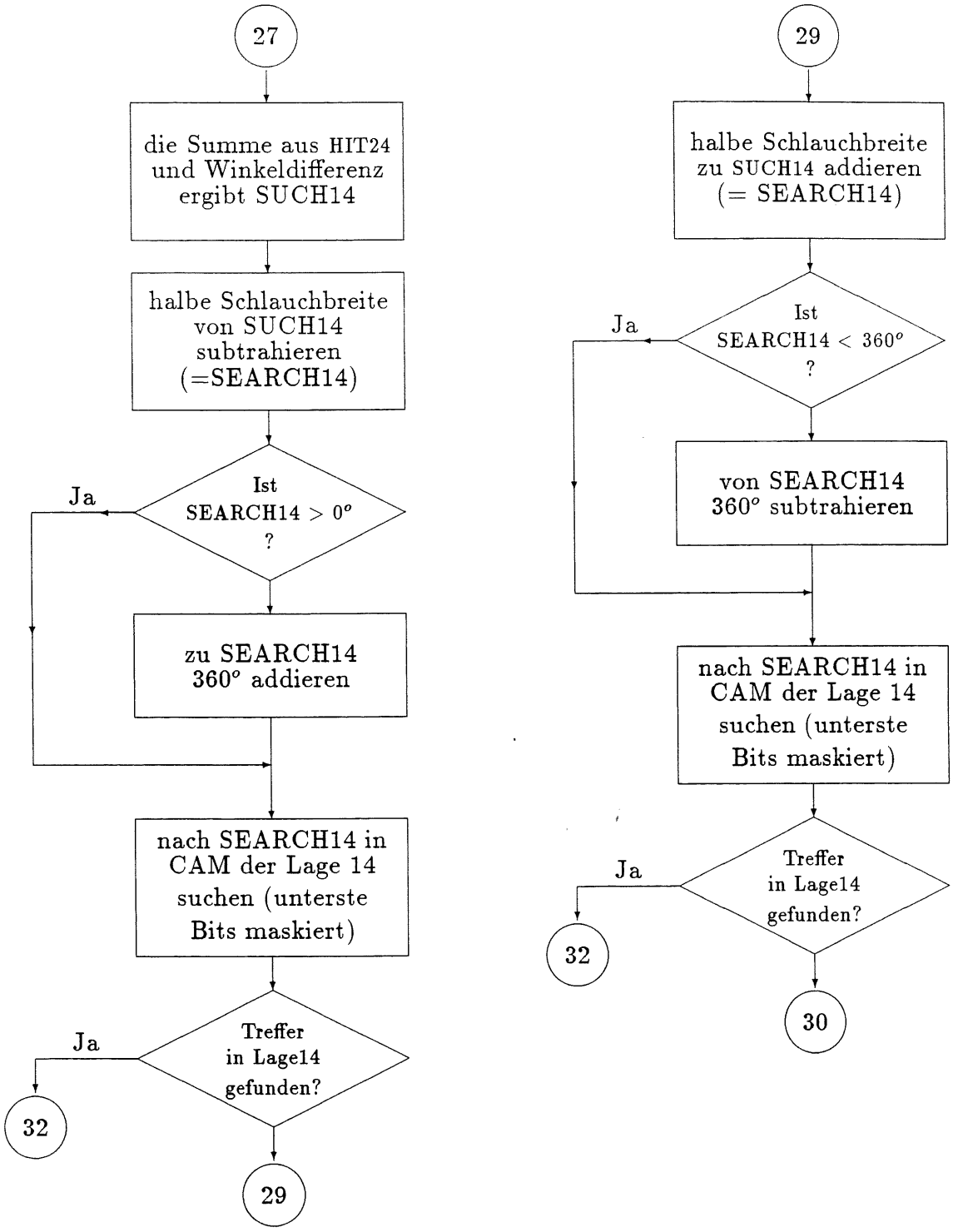


Abbildung 55: Flußdiagramm des verwendeten Programmes (Teil 10)

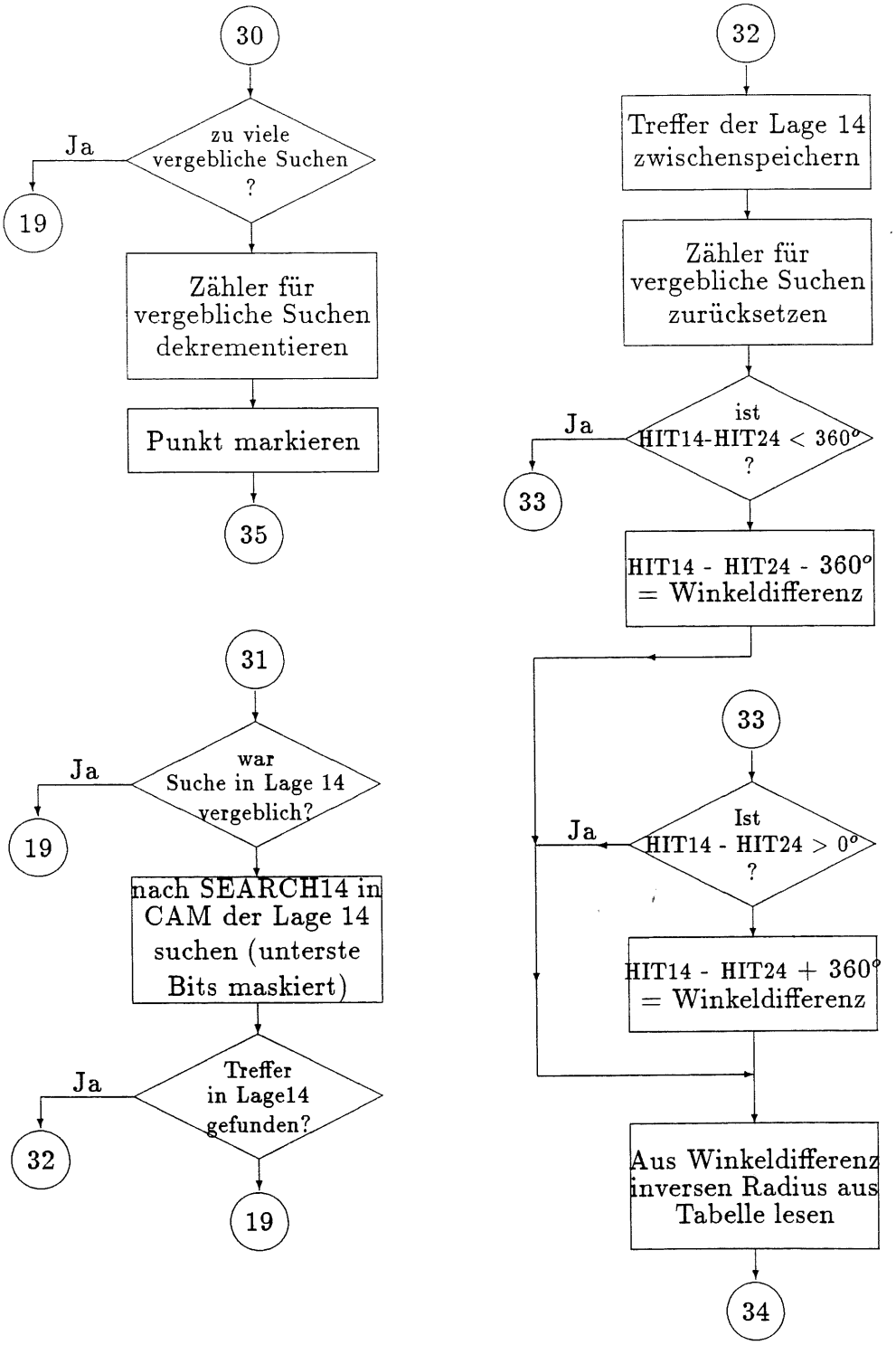


Abbildung 56: Flußdiagramm des verwendeten Programmes (Teil 11)

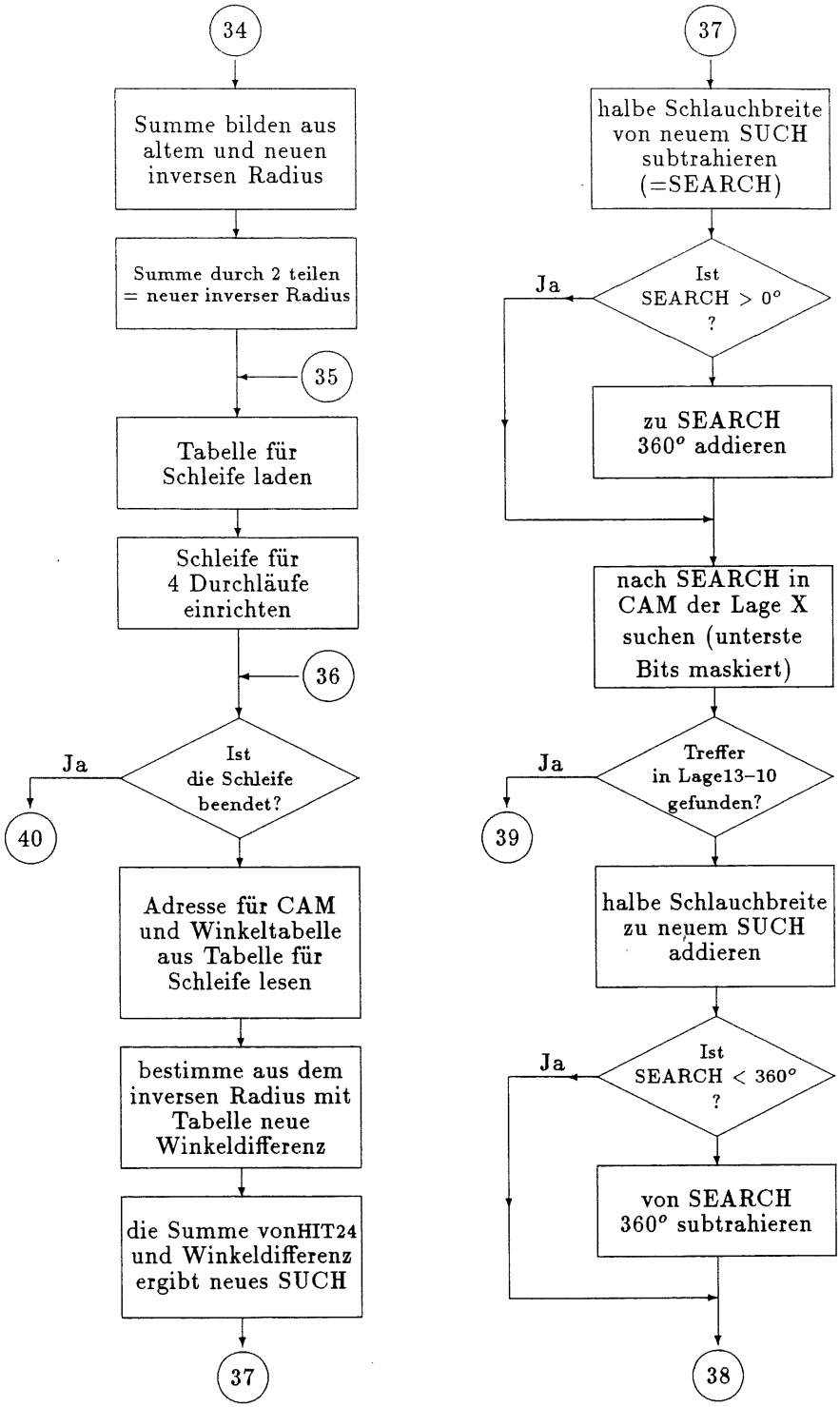


Abbildung 57: Flußdiagramm des verwendeten Programmes (Teil 12)

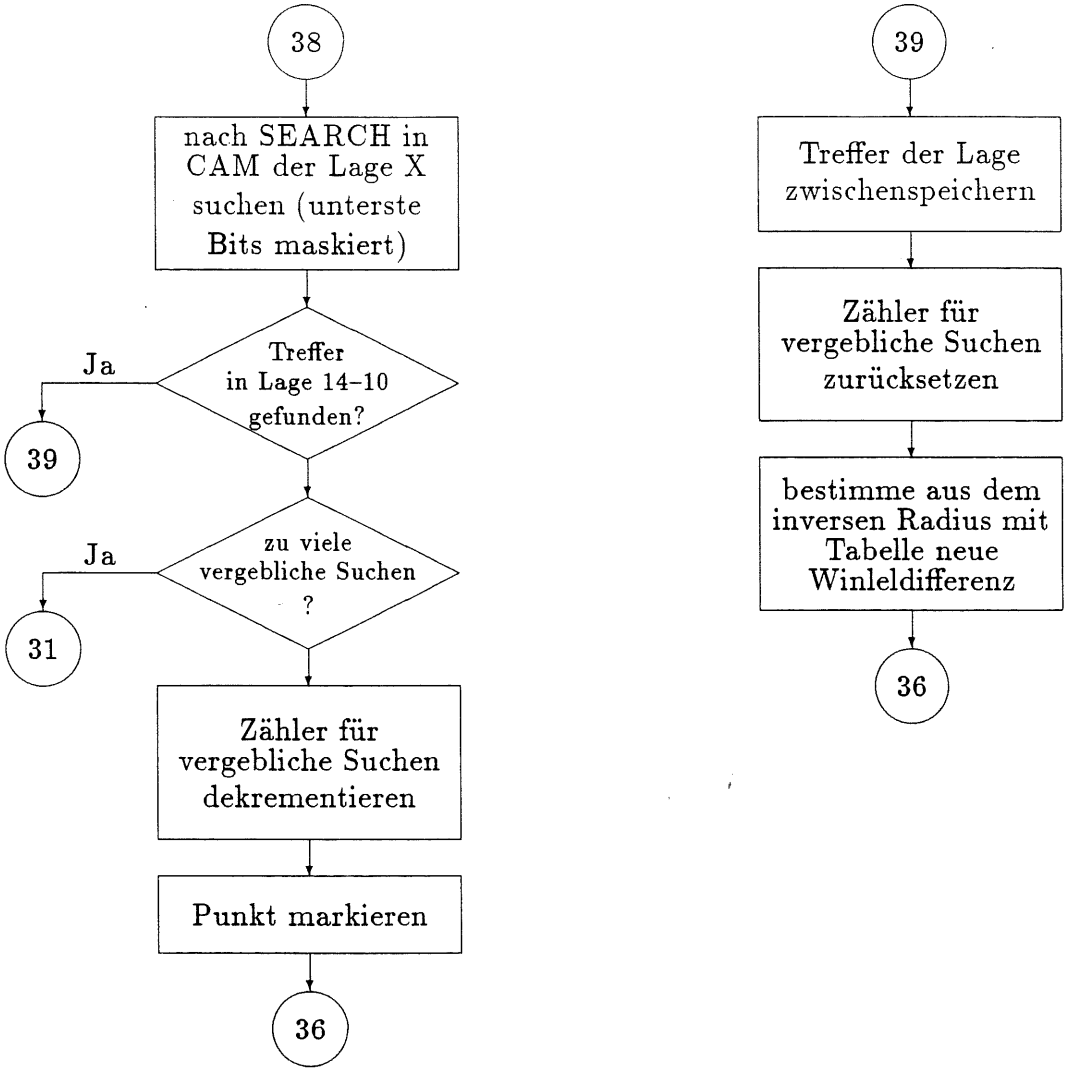


Abbildung 58: Flußdiagramm des verwendeten Programmes (Teil 13)

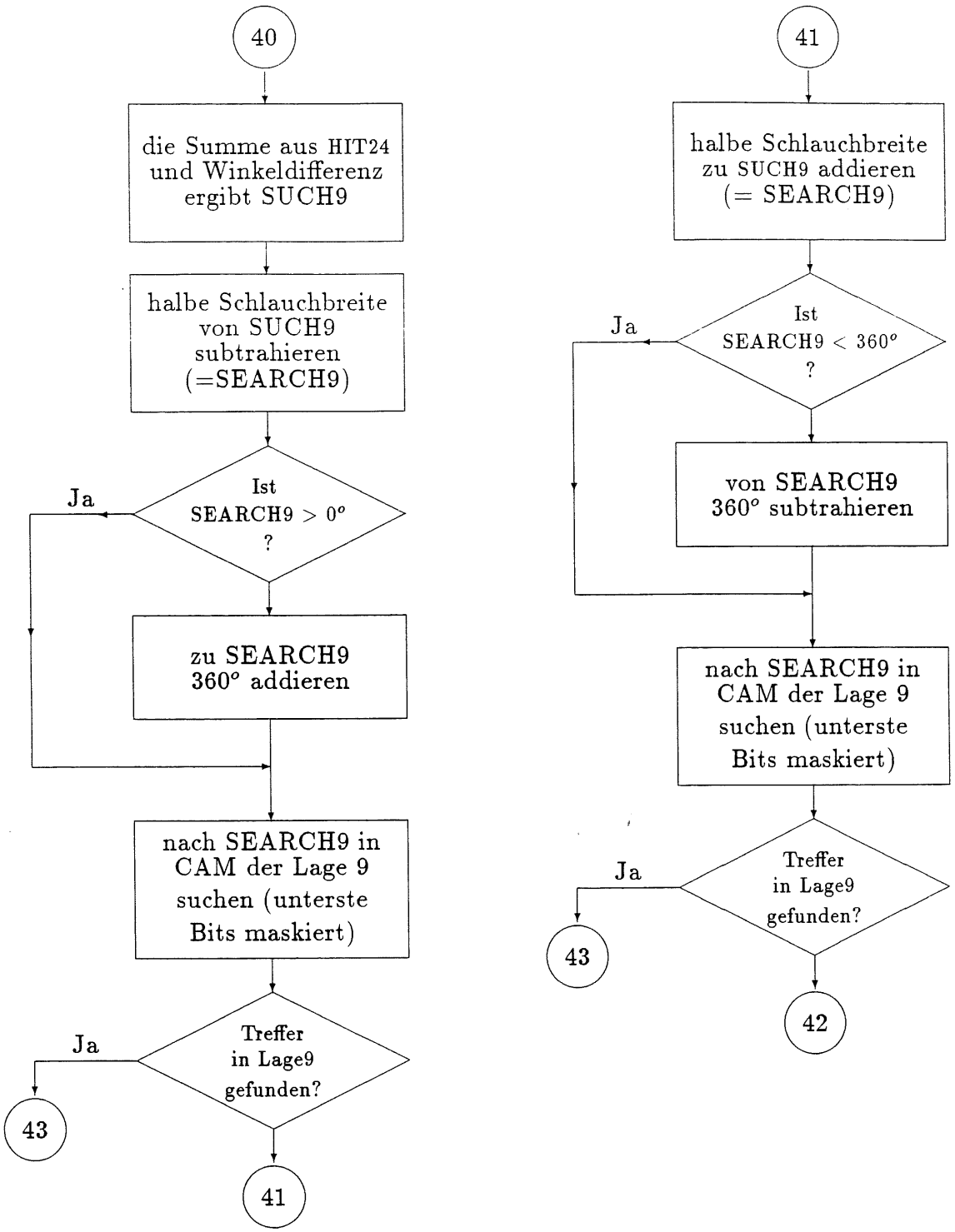


Abbildung 59: Flußdiagramm des verwendeten Programmes (Teil 14)

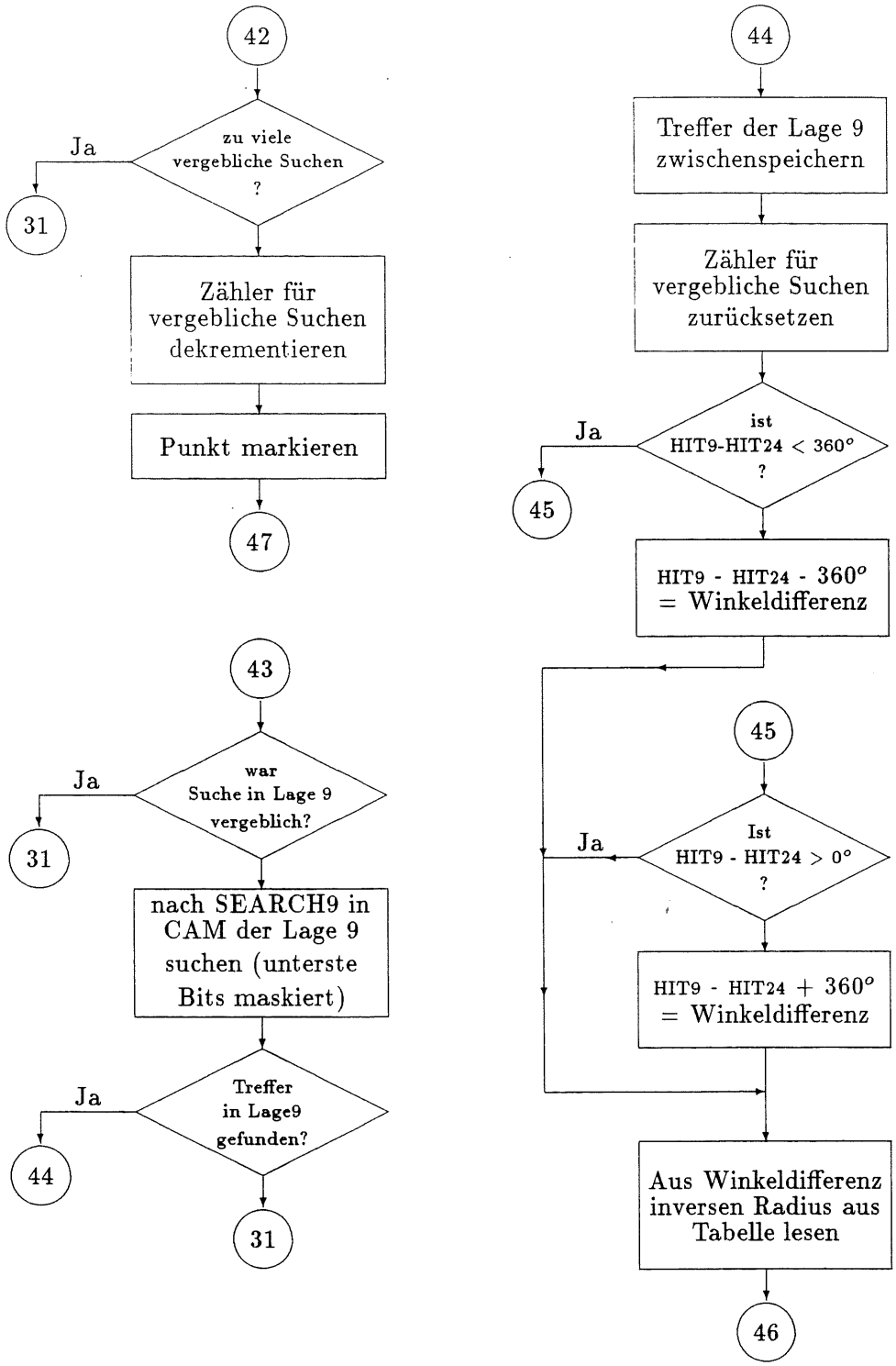


Abbildung 60: Flußdiagramm des verwendeten Programmes (Teil 15)

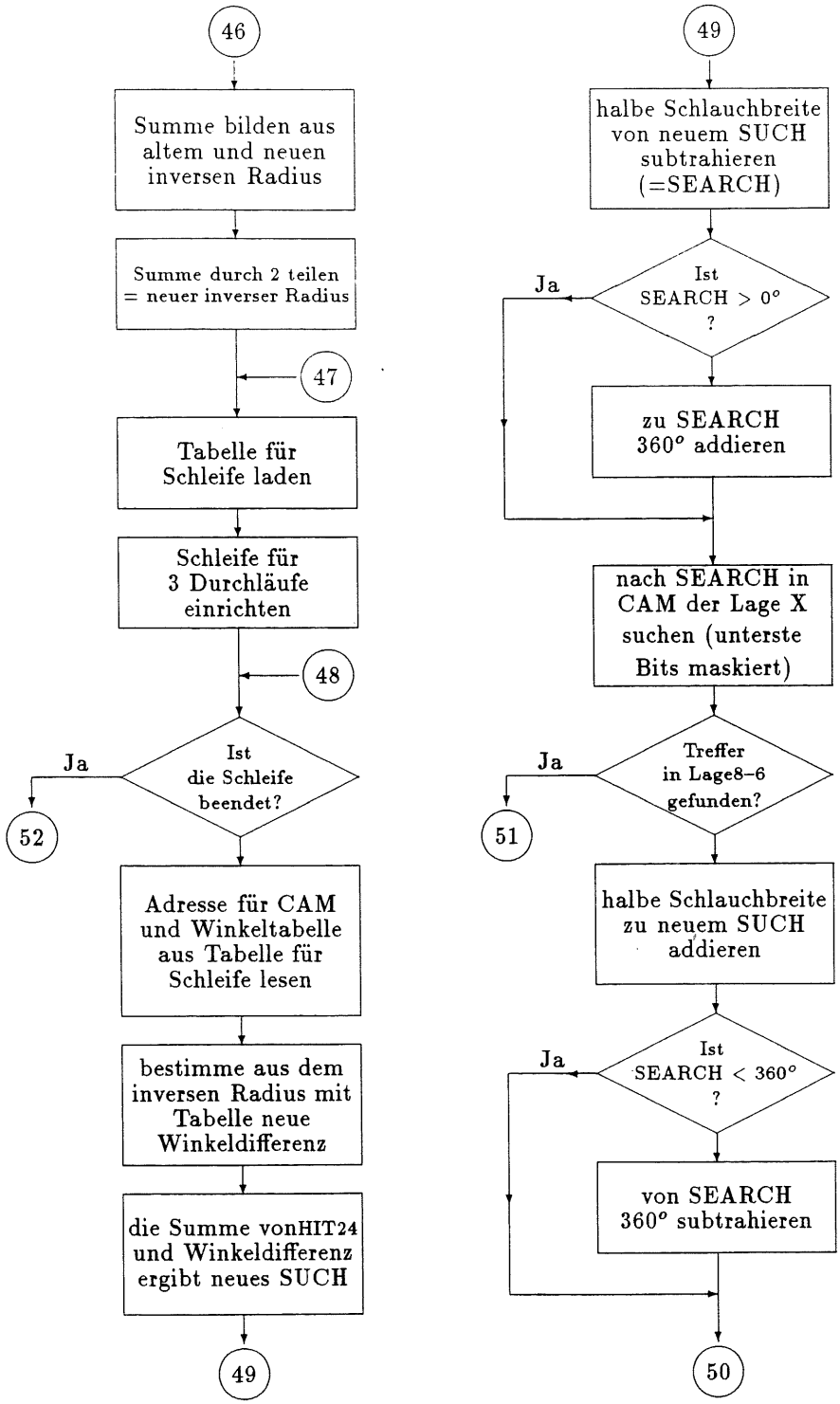


Abbildung 61: Flußdiagramm des verwendeten Programmes (Teil 16)

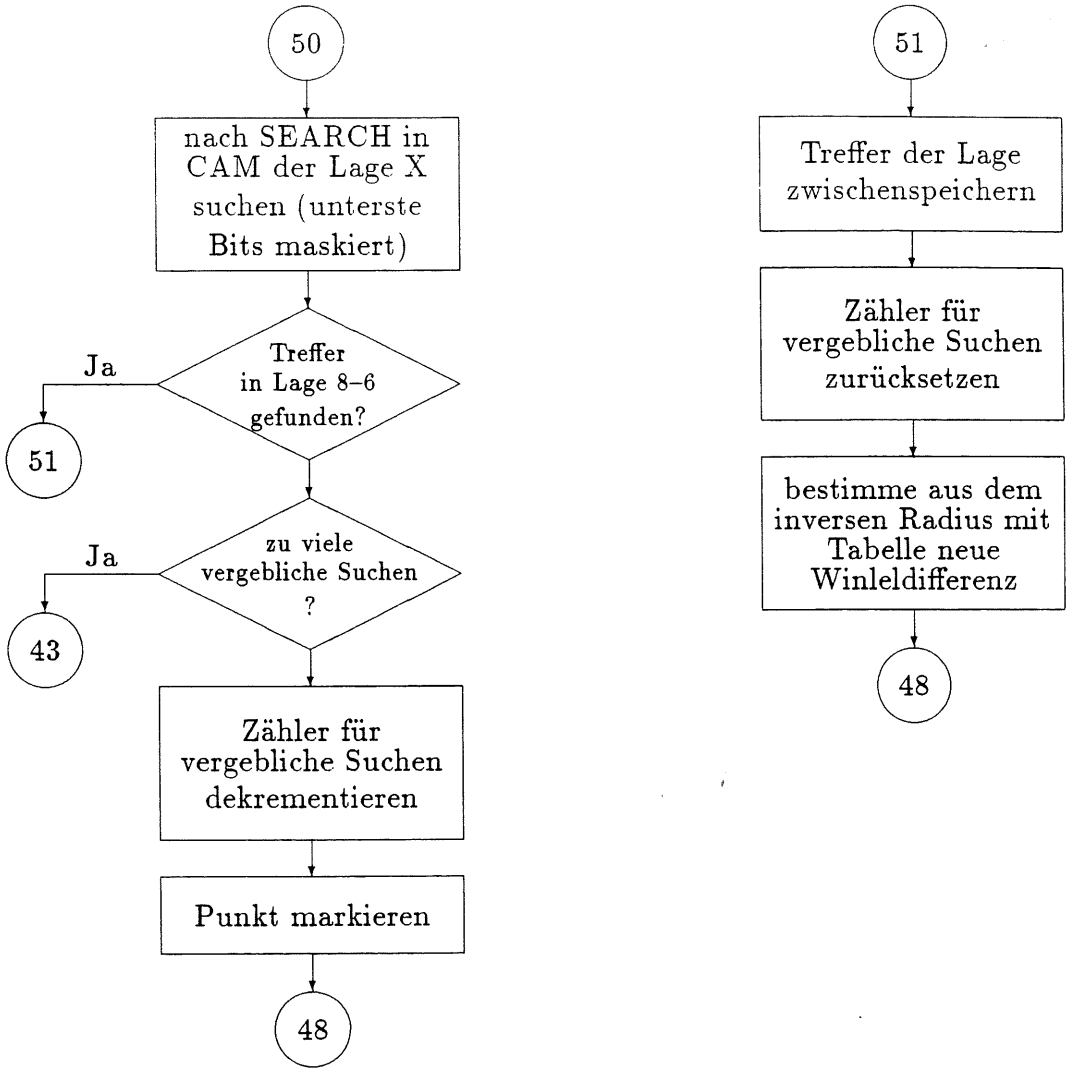


Abbildung 62: Flußdiagramm des verwendeten Programmes (Teil 17)

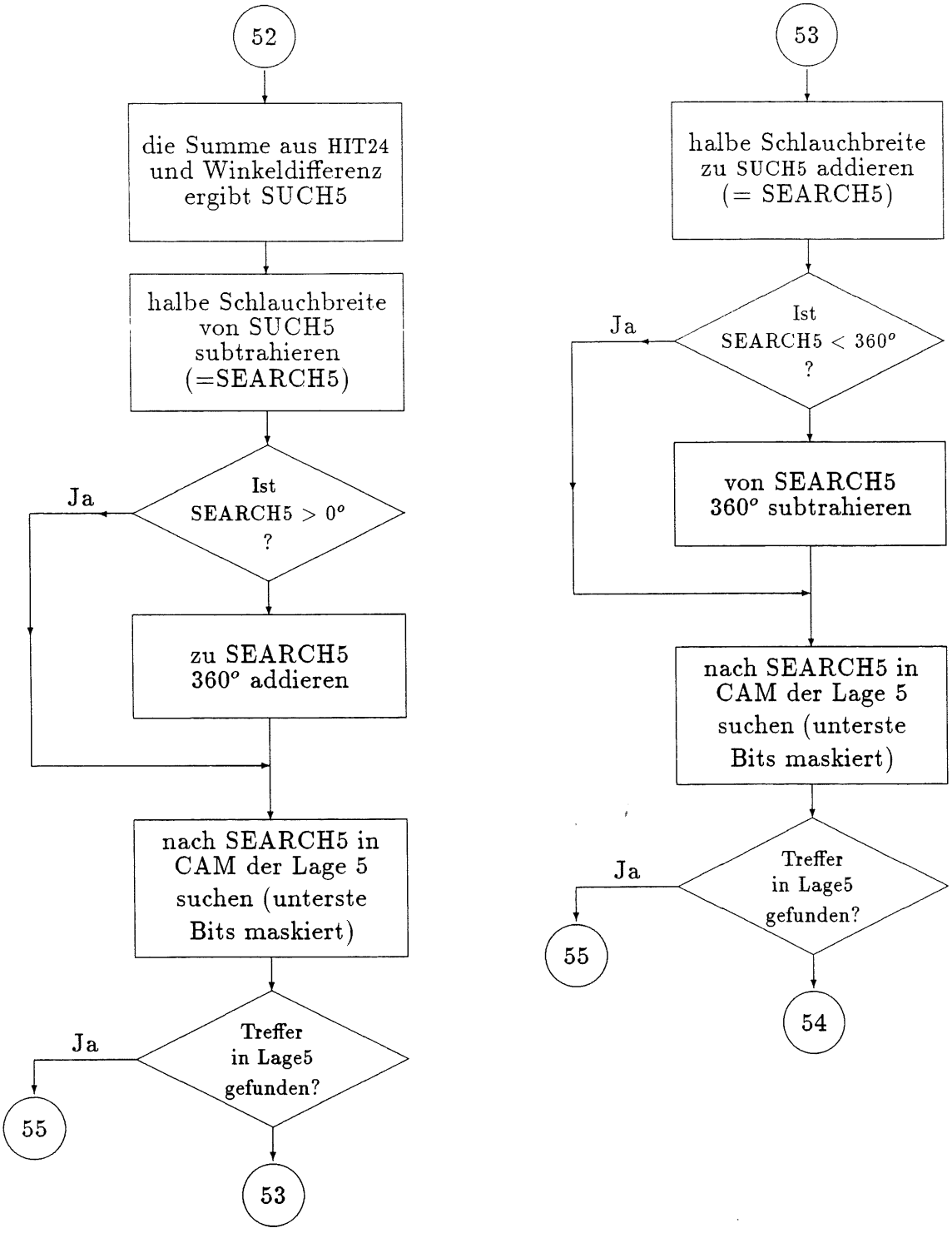


Abbildung 63: Flußdiagramm des verwendeten Programmes (Teil 18)

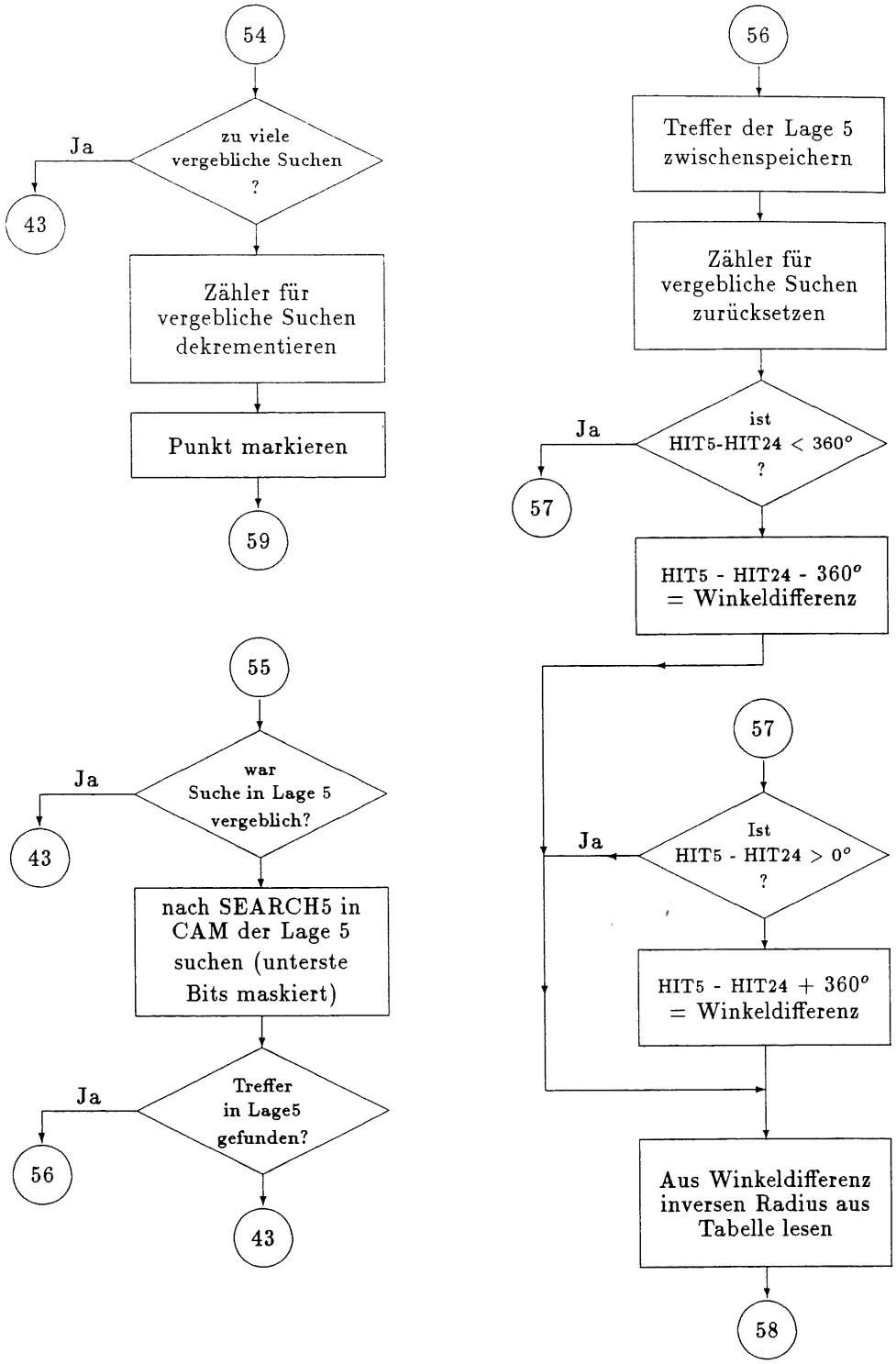


Abbildung 64: Flußdiagramm des verwendeten Programmes (Teil 19)

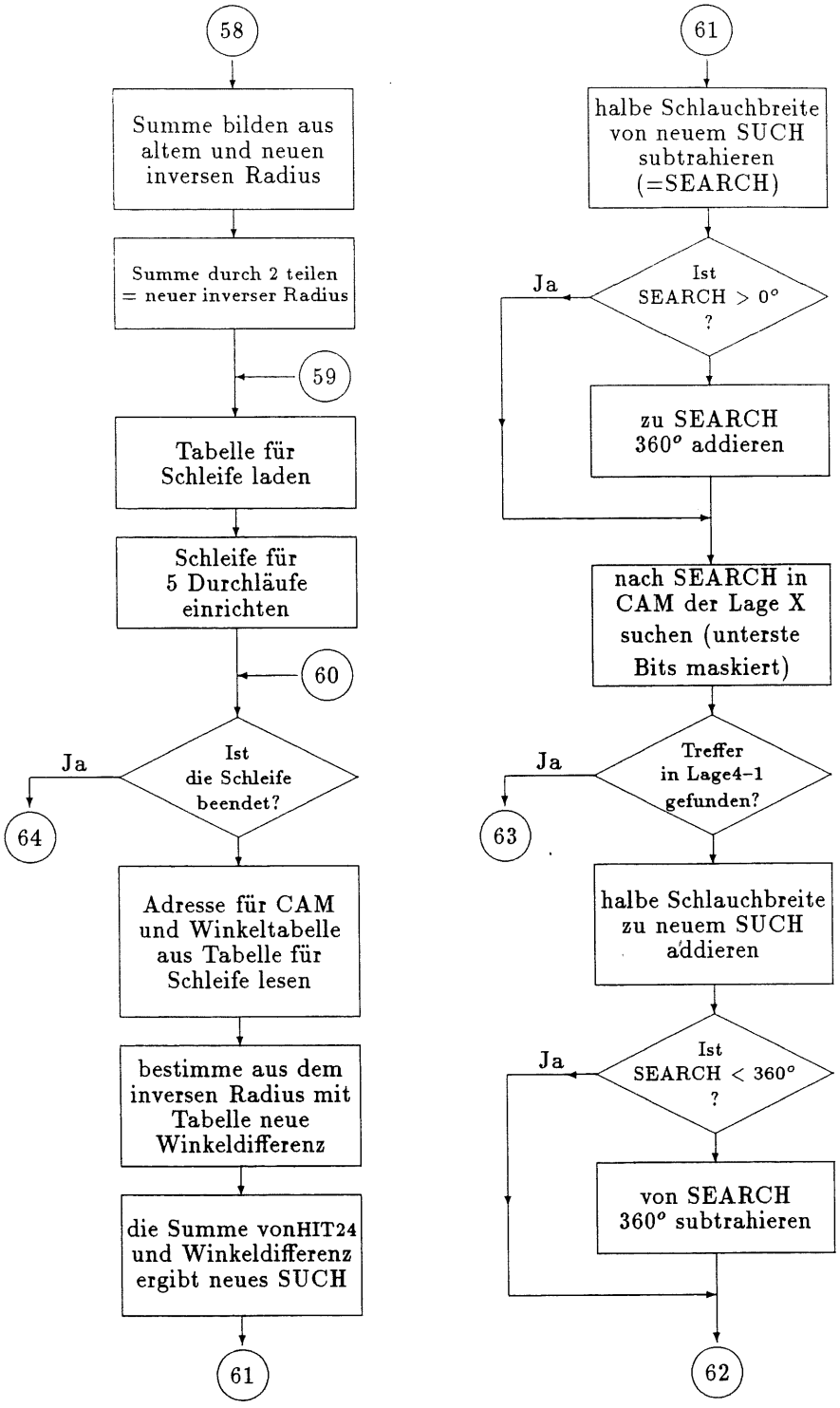


Abbildung 65: Flußdiagramm des verwendeten Programmes (Teil 20)

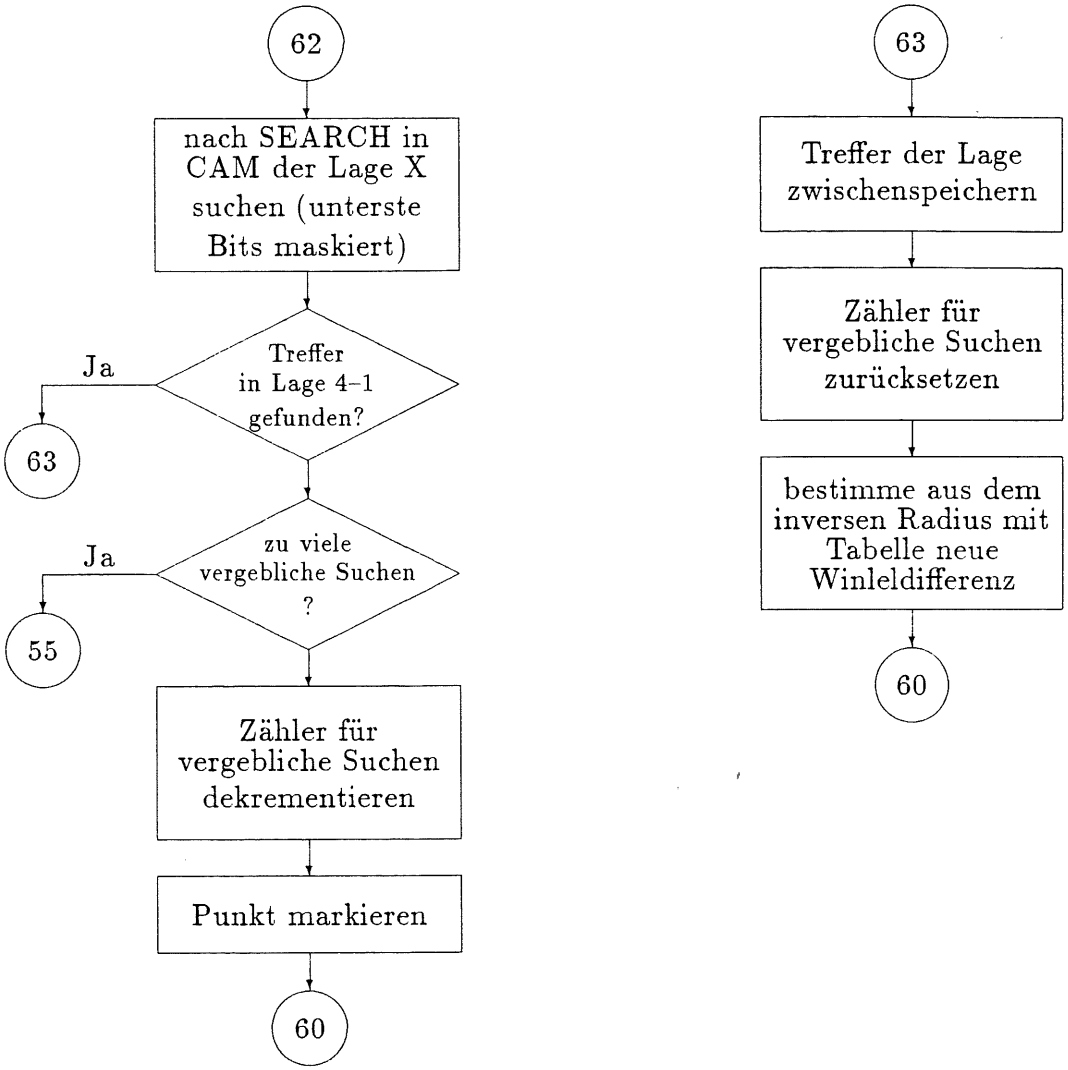


Abbildung 66: Flußdiagramm des verwendeten Programmes (Teil 21)

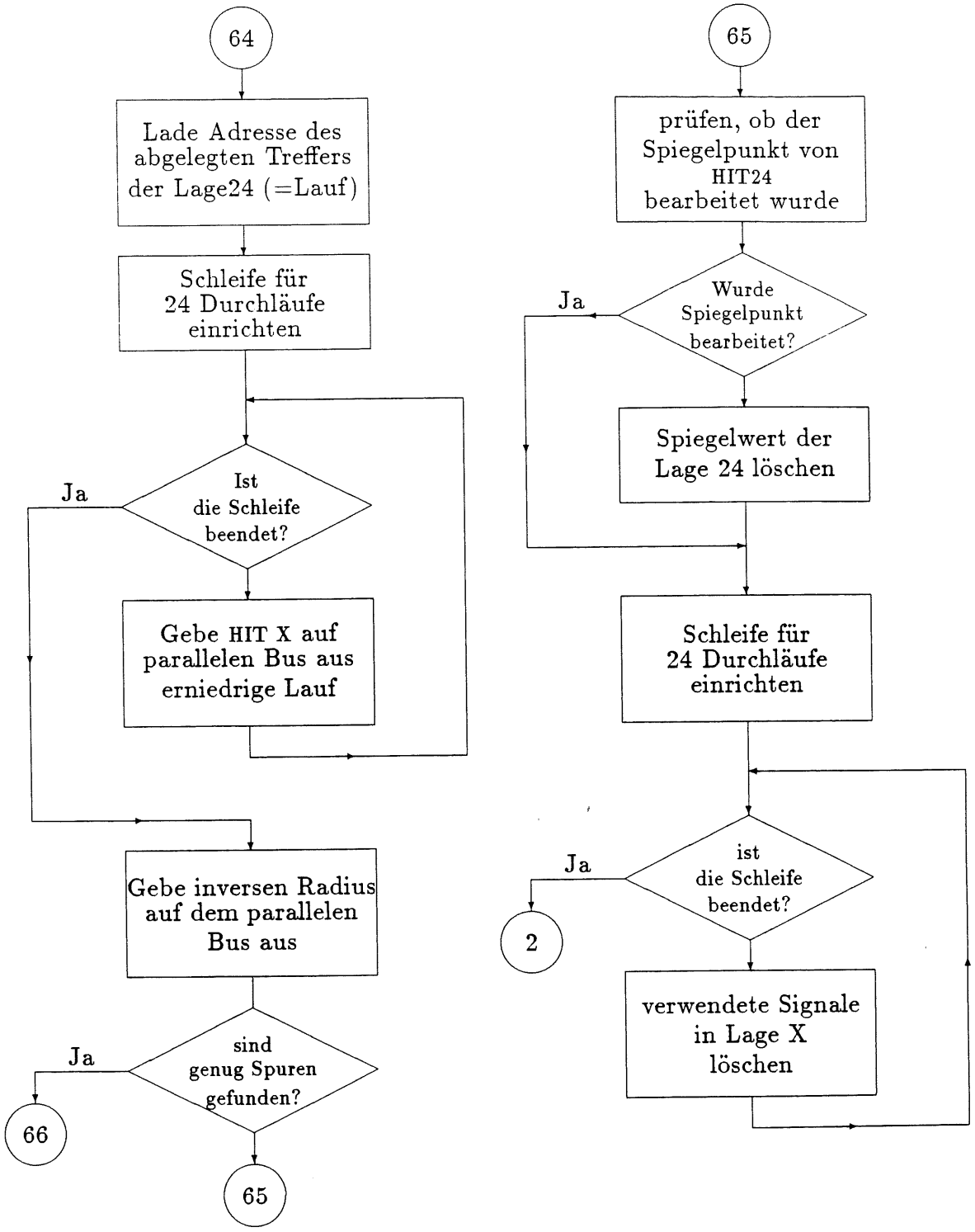


Abbildung 67: Flußdiagramm des verwendeten Programmes (Teil 22)

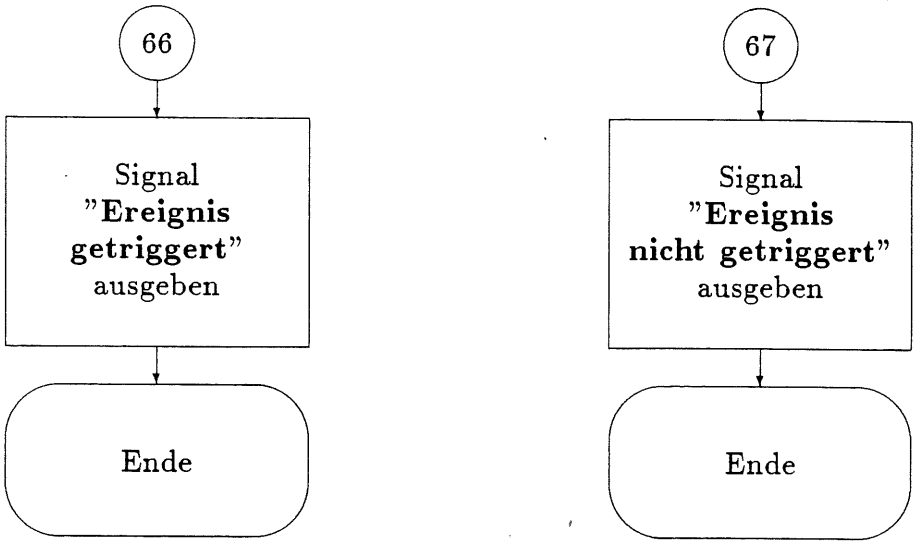


Abbildung 68: Flußdiagramm des verwendeten Programmes (Teil 23)

V.2. Das Assemblerprogramm

Dies ist das für die Simulation verwendete Quellprogramm des untersuchten Triggers auf einem DSP56001 von Motorola.

```
Motorola DSP56000 Macro Cross Assembler  Version 1.00  02-02-88  15:45:17  version9.asm
  Seite 1
Stufe 3 Trigger  Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Adressen für Tabellen und CAMS, Registerinhalte und Konstantendeklaration

5                       PAGE    94,67,0,0,0             ;für IBM LASAR oder STAR
7                       OPT     NOCC,FC


8                 ;
9                 ; Dieses Programm soll MONIKA II auf einem
10                ; DSP56000 von Motorola simulieren!!
11                ; Dieses Programm berechnet nicht Kreisbögen;
12                ; sondern arbeitet mit dem inversen Radius.


13                ;
14                ; Y- Speicheradressen :
15                ;
16      00000231      Treff24  Equ     561
                         ; Adresse für Anzahl Hits in Lage 24
17      00008232      CAMASS23 Equ     33330
                         ; Offset für CAM Lage 23 mit ASSO
18      00008553      CAMASS22 Equ     34131
                         ; Offset für CAM Lage 22 mit ASSO
19      00008874      CAMASS21 Equ     34932
                         ; Offset für CAM Lage 21 mit ASSO
20      00008B95      CAMASS20 Equ     35733
                         ; Offset für CAM Lage 20 mit ASSO
21      00009E5B      CAMASS14 Equ     40539
                         ; Offset für CAM Lage 14 mit ASSO
22      0000AE00      CAMASS9  Equ     44544
                         ; Offset für CAM Lage 9 mit ASSO
23      0000BA84      CAMASS5  Equ     47748
                         ; Offset für CAM Lage 5  mit ASSO
24                ;
25      00000208      RAM24    Equ     520
                         ; Offset für RAM Lage 24
26      00000874      CAM21    Equ     2164
                         ; Offset für CAM Lage 21 ohne ASSO
27      00000B95      CAM20    Equ     2965
                         ; Offset für CAM Lage 20 ohne ASSO
28      00001E5B      CAM14    Equ     7771
                         ; Offset für CAM Lage 14 ohne ASSO
29      00002E00      CAM9     Equ     11776
                         ; Offset für CAM Lage 9 ohne ASSO
30      00003A84      CAM5     Equ     14980
                         ; Offset für CAM Lage 5  ohne ASSO
31                ;
32                ; X- Speicheradressen :
33                ;
34      00000000      Test     Equ     0
                         ; Testbits für Vergeblich
35      000003F3      Tab21r   Equ     1011
                         ; Radiustabelle Lage 21
36      0000040A      Tab20r   Equ     1034
                         ; Radiustabelle Lage 20
37      00000421      Tab14r   Equ     1057
                         ; Radiustabelle Lage 14
38      00000438      Tab9r    Equ     1080
                         ; Radiustabelle Lage 9
39      0000044F      Tab5r    Equ     1103
                         ; Radiustabelle Lage 5
40                ;
41      00000462      Tab21    Equ     1122
                         ; Winkeltabelle Lage 21
42      00000471      Tab20    Equ     1137
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 2
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Adressen für Tabellen und CAMS, Registerinhalte und Konstantendeklaration

```
                                ; Winkeltabelle Lage 20
43      00000596        Tab19   Equ     1430
                                ; Tabelle Lage 19- 15
44      000004CB        Tab14   Equ     1227
                                ; Winkeltabelle Lage 14
45      000005A0        Tab13   Equ     1440
                                ; Winkel Tabelle Lage 13- 10
46      00000516        Tab9    Equ     1302
                                ; Winkeltabelle Lage 9
47      000005A8        Tab8    Equ     1448
                                ; Tabelle Lage 8- 6
48      00000552        Tab5    Equ     1362
                                ; Winkeltabelle Lage 5
49      000005AE        Tab4    Equ     1454
                                ; Tabelle Lage 4- 1
50      000005B6        Tabx    Equ     1462
                                ; Tabelle zum löschen der Hits
51                      ;
52      00000018        P24     Equ     24
                                ; Spurpunkt Lage 24
53      00000017        P23     Equ     23
                                ; Spurpunkt Lage 23
54      00000016        P22     Equ     22
                                ; Spurpunkt Lage 22
55      00000015        P21     Equ     21
                                ; Spurpunkt Lage 21
56      00000014        P20     Equ     20
                                ; Spurpunkt Lage 20
57      00000013        P19     Equ     19
                                ; Spurpunkt Lage 19
58      0000000E        P14     Equ     14
                                ; Spurpunkt Lage 14
59      0000000D        P13     Equ     13
                                ; Spurpunkt Lage 13
60      00000009        P9      Equ     9
                                ; Spurpunkt Lage 9
61      00000008        P8      Equ     8
                                ; Spurpunkt Lage 8
62      00000005        P5      Equ     5
                                ; Spurpunkt Lage 5
63      00000004        P4      Equ     4
                                ; Spurpunkt Lage 4
64                      ;
65      00000019        Push1   Equ     25
                                ; Speicher für CAM- Simulation
66      0000001A        Push2   Equ     26
67      0000001B        Push3   Equ     27
68      0000001C        Push21  Equ     28
69      0000001E        Push20  Equ     30
70      00000020        Push14  Equ     32
71      00000022        Push9   Equ     34
72      00000024        Push5   Equ     36
73      00000026        Trkcnt  Equ     38
                                ; Zähler gefundener Tracks
74                      ; frei ab 39 - 999 und 1508 - $FFFF
75                      ;
76                      ; Sonstige Werte :
77                      ;
78      00000001        Delta   Equ     1
                                ; Konstante für Assotiationen
79      00000320        Maxi    Equ     800
                                ; Maximaler Winkelwert
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 3
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Adressen für Tabellen und CAMS, Registerinhalte und Konstantendeklaration

```
80      00000003    Vergeb  Equ     3
                        ; Anzahl der Fehlsuchen
81      00000024    Winkel  Equ     36
                        ; Testwinkel
82      00008000    Minus   Equ     32768
                        ; Negative 16 Bit Zahl
83      00010000    Trans   Equ     65536
                        ; Transformationswert
84      00FFFFFF    Ok      Equ     $ffffff
                        ; Endemarkierung für Hostrechner(triggered)
85      000AFFE0    No      Equ     $0affe0
                        ; Endemarkierung für Hostrechner(not triggered)
86      00000006    MinTrk  Equ     6
                        ; Mindest Spur Anzahl
87                  ;
88                  ;  verwendete Register
89                  ;
90                  ;R0  Hit- Anzahl in Lage24
91                  ;R1  berechnter Winkel
92                  ;R2  Tabellenadressen für Schleifen
93                  ;R3  Anzahl der vergeblichen Suchen
94                  ;R4  Adresse für Punkte in Schleife
95                  ;R5  Winkeldifferenz
96                  ;R6  Zwischenwinkel
97                  ;R7  Reziproker Radius
98                  ;
99                  ;N0  Pushadresse für CAM
100                 ;N1  CAM- Offset
101                 ;N2  -
102                 ;N3  -
103                 ;N4  -
104                 ;N5  Offset Radientabelle
105                 ;N6  -
106                 ;N7  Offset Winkeltabelle
107                 ;
108                 ;X0  berechneter Winkel
109                 ;X1  Winkel für  Suchbereich
110                 ;
111                 ;Y0  Hit in Lage 24
112                 ;Y1  Maximaler Winkel
113                 ;
114                 ; - = nicht verwendet
```

```
Motorola DSP56000 Macro Cross Assembler  Version 1.00  02-02-88  15:45:17  version9.asm
  Seite 4
Stufe 3 Trigger  Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Setzen der Hardwarevektoren

117                     ;
118                     ;       Setzen des Hardware Reset Vektors
119                     ;
120     P:0000                  org     p:$000
121     P:0000 0AF080           Jmp     Begin
            000040
122                     ;
123                     ;       Stack Error Behandlung
124                     ;
125     P:0002                  org     p:$2
126     P:0002 0C0002  enden    Jmp     <enden
127                     ;
128                     ;       Host receive data vektor
129                     ;
130     P:0020                  org     p:$20
131     P:0020 0C0020  receive  jmp     <receive
132                     ;
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 5
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programmanfang

```
135                     ;       Programmbegin bei p:$0040
136                     ;
137     P:0040                  org     p:$0040
138     P:0040 08F4BE  Begin    Movep           #0,X:<<$FFFE
            000000              ; Keine Waitstates für Memory
139     P:0042 68F000           Move                    Y:Treff24,R0
            000231              ; Anzahl der Hits in Lage 24
140     P:0044 61F400           Move            #>MinTrk,R1
            000006
141     P:0046 612600           Move            R1,X:Trkcnt
                                ; Noch Mintrk Spuren finden
142     P:0047 45F400           Move            #>Delta,X1
            000001              ; Offset für das Suchen
143     P:0049 612600           Move            R1,X:Trkcnt
                                ; Noch Mintrk Spuren finden
144     P:004A 47F400           Move            #>Maxi,Y1
            000320              ; Maximaler Winkelwert
145     P:004C 220E00           Move            R0,A
146     P:004D 200003           Tst     A
                                ; Sind in Lage 24 Hits?
147     P:004E 0EA2B1           JEQ     <End24
149                     ;
150                     ; Lage 24 Schleife
151                     ;
152     P:004F 06D000           Do      R0,Loop24
            0002B1              ; Schleife aller Hits in Lage 24
153     P:0051 044EBF           Move            LC,A
                                ; Lade Schleifenzähler
154     P:0052 57A600           Move            X:Trkcnt,B
                                ; Hole fehlende Spuranzahl
155     P:0053 200005           Cmp     B,A
156     P:0054 0AF0A1           JGE     Weiter
            000059
157     P:0056 00008C           Enddo
158     P:0057 0AF080           Jmp     FIN
            0002BB
159     P:0059 70F400  Weiter   Move            #RAM24,N0
            000208
160     P:005B 370000           Move            #0,R7
161     P:005C 330300           Move            #Vergeb,R3
                                ; Zähler der vergeblichen Versuche
162     P:005D 5FE813           Clr     A               Y:(R0+N0),B
                                ; Hit aus Lage 24 holen
163     P:005E 045010           Lua     (R0)-,R0
164     P:005F 560000           Move            A,X:Test
                                ; Testbits Zuruecksetzen
165     P:0060 71F400           Move            #CAMASS23,N1
            008232              ; Offset für CAM Lage 23 mit ASSO
166     P:0062 21FE00           Move            B,N6
                                ; Hit in Lage 24
167                     ;
168                     ; Lage 23 Suchen
169                     ;
170     P:0063 21E600           Move            B,Y0
                                ; Sichern des Hits in Lage 24
171     P:0064 46186C           Sub     X1,B    Y0,X:P24
                                ; Speichern des Spurpunktes
172
                                ; Verbesserung der Effizienz der CAMS
173     P:0065 0E1068           JGE     <Plus23
174     P:0066 0A0020           Bset    #0,x:Test
175     P:0067 200078           ADD     Y1,B
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 6
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
176     P:0068 21F100  Plus23   Move          B,R1
                        ; 1. Erwarteter Hit in Lage 23
177     P:0069 0D02EB           Jsr     <CAMM
                        ; Hit aus Lage 23 holen (mit ASSO)
178     P:006A 20000B           Tst     B
                        ; Testen ob Suche Erfolgreich
179     P:006B 0E107A           JGE     <Hit23
                        ; Wenn Erfolgreich Sprung
180                 ;
181                 ;   1. Mal Vergebliche Suche in Lage 23
182                 ;
183     P:006C 20CF00           Move            Y0,B
                        ; Hit Lage 24
184     P:006D 200068           Add     X1,B
                        ; Verbesserung der Effizienz der CAMS
185     P:006E 20007D           CMP     Y1,B
                        ; Winkel Testen
186     P:006F 0E9072           JLT     <Ok23
                        ; Wenn Winkel Positiv Sprung
187     P:0070 0A0020           Bset    #0,x:Test
188     P:0071 20007C           Sub     Y1,B
                        ; Winkel Positiv machen
189     P:0072 21F100  Ok23     Move            B,R1
                        ; 2. Erwarteter Hit in Lage 23
190     P:0073 0D02EB           Jsr     <CAMM
                        ; Hit aus Lage 23 holen (mit ASSO)
191     P:0074 20000B           Tst     B
                        ; Testen ob Suche Erfolgreich
192     P:0075 0E107A           JGE     <Hit23
                        ; Wenn Erfolgreich Sprung
193                 ;
194                 ;   Vergeblich
195                 ;
196     P:0076 471700           Move            Y1,X:P23
                        ; Vergebliche Suche
197     P:0077 045313           Lua     (R3)-,R3
                        ; Zähler Erniedrigen
198     P:0078 20CF00           Move            Y0,B
199     P:0079 0C007C           Jmp     <Hitv23
200                 ;
201                 ;   Lage 22 Suchen
202                 ;
203     P:007A 571700  Hit23    Move            B,X:P23
                        ; Speichern des Spurpunktes
204     P:007B 330300           Move            #Vergeb,R3
                        ; Zähler der vergeblichen Versuche
205     P:007C 71F400  Hitv23   Move            #CAMASS22,N1
            008553          ; Offset für CAM Lage 22 mit ASSO
206     P:007E 20CF00           Move            Y0,B
                        ; Holen des Hits in Lage 24
207     P:007F 20006C           Sub     X1,B
                        ; Verbesserung der Effizienz der CAMS
208     P:0080 0E1083           JGE     <Plus22
209     P:0081 0A0020           Bset    #0,x:Test
210     P:0082 200078           ADD     Y1,B
211     P:0083 21F100  Plus22   Move            B,R1
                        ; 1. Erwarteter Hit in Lage 22
212     P:0084 0D02EB           Jsr     <CAMM
                        ; Hit aus Lage 22 holen (mit ASSO)
213     P:0085 20000B           Tst     B
                        ; Testen ob Suche Erfolgreich
214     P:0086 0E1095           JGE     <Hit22
```

```
                        ; Wenn Erfolgreich Sprung
215                 ;
216                 ;   1. Mal Vergebliche Suche in Lage 22
217                 ;
218     P:0087 20CF00           Move            Y0,B
                        ; Hit Lage 24
219     P:0088 200068           Add     X1,B
                        ; Verbesserung der Effizienz der CAMS
220     P:0089 20007D           CMP     Y1,B
                        ; Winkel Testen
221     P:008A 0E908D           JLT     <Ok22
                        ; Wenn Winkel Positiv Sprung
222     P:008B 0A0020           Bset    #0,x:Test
223     P:008C 20007C           Sub     Y1,B
                        ; Winkel Positiv machen
224     P:008D 21F100  Ok22     Move            B,R1
                        ; 2. Erwarteter Hit in Lage 22
225     P:008E 0D02EB           Jsr     <CAMM
                        ; Hit aus Lage 22 holen (mit ASSO)
226     P:008F 20000B           Tst     B
                        ; Testen ob Suche Erfolgreich
227     P:0090 0E1095           JGE     <Hit22
                        ; Wenn Erfolgreich Sprung
228                 ;
229                 ;  Vergeblich
230                 ;
231     P:0091 471600           Move            Y1,X:P22
                        ; Vergebliche Suche
232     P:0092 045313           Lua     (R3)-,R3
                        ; Zähler Erniedrigen
233     P:0093 20CF00           Move            Y0,B
234     P:0094 0C0098           Jmp     <Hitv22
235                 ;
236                 ;   Lage 21 Suchen
237                 ;
238     P:0095 571600  Hit22    Move            B,X:P22
                        ; Speichern des Spurpunktes
239     P:0096 330300           Move            #Vergeb,R3
                        ; Zähler der vergeblichen Versuche
240     P:0097 0A0015           BClr    #21,X:Test
                        ; Testbit löschen
241     P:0098 71F400  Hitv22   Move            #CAMASS21,N1
            008874          ; Offset für CAM Lage 21 mit ASSO
242     P:009A 20CF00           Move            Y0,B
                        ; Holen des Hits in Lage 24
243     P:009B 20006C           Sub     X1,B
                        ; Verbesserung der Effizienz der CAMS
244     P:009C 0E109F           JGE     <Plus21
245     P:009D 0A0020           Bset    #0,x:Test
246     P:009E 200078           ADD     Y1,B
247     P:009F 21F100  Plus21   Move            B,R1
                        ; 1. Erwarteter Hit in Lage 21
248     P:00A0 0D02EB           Jsr     <CAMM
                        ; Hit aus Lage 21 holen (mit ASSO)
249     P:00A1 20000B           Tst     B
                        ; Testen ob Suche Erfolgreich
250     P:00A2 0E10B9           JGE     <Hit21
                        ; Wenn Erfolgreich Sprung
251                 ;
252                 ;   1. Mal Vergebliche Suche in Lage 21
253                 ;
254     P:00A3 20CF00           Move            Y0,B
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 8
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
                        ; Hit Lage 24
255     P:00A4 200068           Add     X1,B
                        ; Verbesserung der Effizienz der CAMS
256     P:00A5 20007D           CMP     Y1,B
                        ; Winkel Testen
257     P:00A6 0E90A9           JLT     <Ok21
                        ; Wenn Winkel Positiv Sprung
258     P:00A7 0A0020           Bset    #0,x:Test
259     P:00A8 20007C           Sub     Y1,B
                        ; Winkel Positiv machen
260     P:00A9 21F100  Ok21     Move            B,R1
                        ; 2. Erwarteter Hit in Lage 21
261     P:00AA 0D02EB           Jsr     <CAMM
                        ; Hit aus Lage 21 holen (mit ASSO)
262     P:00AB 20000B           Tst     B
                        ; Testen ob Suche Erfolgreich
263     P:00AC 0E10B9           JGE     <Hit21
                        ; Wenn Erfolgreich Sprung
264                     ;
265                     ;  Vergeblich
266                     ;
267     P:00AD 471500           Move            Y1,X:P21
                        ; Vergebliche Suche
268     P:00AE 045313           Lua     (R3)-,R3
                        ; Zähler Erniedrigen
269     P:00AF 20CF00           Move            Y0,B
270     P:00B0 0C00CE           Jmp     <Point21
271                     ;
272                     ;   Einsprung in Lage 21 nach vergeblicher Suche
273                     ;
274     P:00B1 71F400  Lage21   Move            #CAM21,N1
            000874          ; Cam Offset ohne Asso
275     P:00B3 0A00B5           Jset    #21,X:Test,End24
            0002B1          ; Neuer Wert in Lage 24
276     P:00B5 381C00           Move            #Push21,N0
277     P:00B6 0D02BF           Jsr     <CAM
                        ; Hit aus Lage 21 holen (mit ASSO)
278     P:00B7 20000B           Tst     B
279     P:00B8 0E92B1           JLT     <End24
280     P:00B9 571500  Hit21    Move            B,X:P21
                        ; Speichern des Spurpunktes
281     P:00BA 330300           Move            #Vergeb,R3
                        ; Zähler der vergeblichen Versuche
282     P:00BB 20005C           Sub     Y0,B
                        ; Winkeldifferenz bilden
283     P:00BC 0A0080           Jclr    #0,X:Test,Null21
            0000C9          ; Wurden Nullgrad ueberschritten
284     P:00BE 56F400           Move            #>Winkel,A
            000024          ; Maximalwert
285     P:00C0 20000D           Cmp     A,B
                        ; Ist Winkeldifferenz größer
286     P:00C1 0EF0C5           JLE     <Null21a
287     P:00C2 20007C           Sub     Y1,B
                        ; Winkel zu gross
288     P:00C3 20003E           Neg     B
                        ; Maximalwinkel - Winkeldifferenz
289     P:00C4 0C00C9           Jmp     <Null21
290     P:00C5 200036  Null21a  Neg     A
                        ; Winkel zu klein (negativ)
291     P:00C6 20000D           Cmp     A,B
                        ; Maximalwinkel + Winkeldifferenz
292     P:00C7 0E10C9           JGE     <Null21
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm Seite 9
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation Programm

```
293     P:00C8 200078          Add     Y1,B
294     P:00C9 21F500  Null21  Move            B,R5
295     P:00CA 75F400          Move            #Tab21r,N5
            0003F3          ; Offset Radientabelle
296     P:00CC 000000          Nop
297     P:00CD 67ED00          Move            X:(R5+N5),R7
                           ; Inverser Radius suchen
298                ;
299                ;   Lage 20 Suchen
300                ;
301     P:00CE 77F400  Point21 Move            #Tab20,N7
            000471          ; Offset Winkeltabelle
302     P:00D0 71F400          Move            #CAMASS20,N1
            008B95          ; Offset CAM20 mit ASSO
303     P:00D2 57EF00          Move            X:(R7+N7),B
                           ; Berechnen von neuen Winkel
304     P:00D3 200058          Add     Y0,B
305     P:00D4 21E400          Move            B,X0
                           ; Winkel sichern
306     P:00D5 20006C          Sub     X1,B
                           ; Verbesserung der Effizienz der CAMS
307     P:00D6 0E10D9          JGE     <Plus20
308     P:00D7 200078          Add     Y1,B
309     P:00D8 0A0020          Bset    #0,x:Test
310     P:00D9 21F100  Plus20  Move            B,R1
                           ; 1. Erwarteter Hit in Lage 20
311     P:00DA 381E00          Move            #Push20,N0
312     P:00DB 0D02BF          Jsr     <CAM
                           ; Hit aus Lage 20 holen (mit ASSO)
313     P:00DC 20000B          Tst     B
                           ; Testen ob Suche Erfolgreich
314     P:00DD 0E10F9          JGE     <Hit20
                           ; Wenn Erfolgreich Sprung
315                ;
316                ;   1. Mal Vergebliche Suche in Lage 20
317                ;
318     P:00DE 200049          Tfr     X0,B
                           ; gesuchter Winkel Lage 20
319     P:00DF 200068          Add     X1,B
                           ; Verbesserung der Effizienz der CAMS
320     P:00E0 20007D          CMP     Y1,B
                           ; Winkel Testen
321     P:00E1 0E90E4          JLT     <Ok20
                           ; Wenn Winkel Positiv Sprung
322     P:00E2 20007C          Sub     Y1,B
                           ; Winkel Positiv machen
323     P:00E3 0A0020          Bset    #0,x:Test
324     P:00E4 21F100  Ok20    Move            B,R1
                           ; 2. Erwarteter Hit in Lage 20
325     P:00E5 381E00          Move            #Push20,N0
326     P:00E6 0D02BF          Jsr     <CAM
                           ; Hit aus Lage 20 holen (mit ASSO)
327     P:00E7 20000B          Tst     B
                           ; Testen ob Suche Erfolgreich
328     P:00E8 0E10F9          JGE     <Hit20
                           ; Wenn Erfolgreich Sprung
329                ;
330                ;   Vergebliche Suche in Lage 20
331                ;
332
333     P:00E9 226F00          Move            R3,B
334     P:00EA 20000B          Tst     B
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 10
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
335     P:00EB 0EF0B1          JLE    <Lage21
                        ; Neuer Wert in Lage 21 suchen
336     P:00EC 045313          Lua    (R3)-,R3
                        ; Zähler erniedrigen
337     P:00ED 471459          Tfr    Y0,B     Y1,X:P20
                        ; Vergebliche Suche
338     P:00EE 0A0034          BSet   #20,X:Test
                        ;
339     P:00EF 0C0119          Jmp    <Point20
340             ;
341             ;   Einsprung in Lage 20 nach vergeblicher Suche
342             ;
343     P:00F0 00008C  Lage20a  Enddo
344     P:00F1 71F400  Lage20   Move           #CAM20,N1
            000B95          ; CAM Offset ohne Asso
345     P:00F3 0A00B4          Jset   #20,X:Test,Lage21
            0000B1          ; Neuer Wert in Lage 21
346     P:00F5 381E00          Move           #Push20,N0
347     P:00F6 0D02BF          Jsr    <CAM
                        ; Hit aus Lage 22 holen (mit ASSO)
348     P:00F7 20000B          Tst    B
349     P:00F8 0E90B1          JLT    <Lage21
350     P:00F9 571400  Hit20   Move           B,X:P20
                        ; Speichern des Spurpunktes
351     P:00FA 330300          Move           #Vergeb,R3
                        ; Zähler der vergeblichen Versuche
352     P:00FB 0A0014          Bclr   #20,x:Test
353     P:00FC 21E45C          Sub    Y0,B     B,X0
                        ; Sichern des Punktes , Winkeldifferenz bilden
354     P:00FD 0A0080          Jclr   #0,X:Test,Null20
            00010A          ; Wurde Nullgrad Überschritten
355     P:00FF 56F400          Move           #>Winkel,A
            000024          ; Ist Winkeldifferenz größer
356     P:0101 20000D          Cmp    A,B
                        ;    Maximalwert
357     P:0102 0EF106          JLE    <Null20a
358     P:0103 20007C          Sub    Y1,B
                        ; Winkel zu gross
359     P:0104 20003E          Neg    B
                        ; Maximalwinkel - Winkeldifferenz
360     P:0105 0C010A          Jmp    <Null20
361     P:0106 200036  Null20a  Neg    A
                        ; Winkel zu klein (negativ)
362     P:0107 20000D          Cmp    A,B
                        ; Maximalwinkel + Winkeldifferenz
363     P:0108 0E110A          JGE    <Null20
364     P:0109 200078          Add    Y1,B
365     P:010A 21F500  Null20   Move           B,R5
                        ;
366     P:010B 75F400          Move           #Tab20r,N5
            00040A          ; Offset Radientabelle
367     P:010D 22EF00          Move           R7,B
                        ; Alter Inverser Radius
368     P:010E 56F400          Move           #Minus,A
            008000
369     P:0110 20000D          Cmp    A,B
370     P:0111 0E9115          JLT    <Nega20
                        ; Ist der alte Winkel negativ,
371     P:0112 56F400          Move           #Trans,A
            010000          ; dann soll es B auch sein
372     P:0114 200018          Add    A,B
373     P:0115 56ED00  Nega20   Move           X:(R5+N5),A
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 11
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
                        ; Inverser Radius suchen
374     P:0116 200018           Add     A,B
                        ; Mittelwert bilden
375     P:0117 20002B           Lsr     B
376     P:0118 21F700           Move            B,R7
                        ; Mittelwert ablegen
377                 ;
378                 ;
379                 ;   Lage 19- 15 Suchen
380                 ;
381     P:0119 62F400 Point20   Move            #Tab19,R2
            000596
382     P:011B 341300           Move            #P19,R4
383     P:011C 060580           Do      #5,Loop19
            00013D
384     P:011E 77DA00           Move            X:(R2)+,N7
                        ; Offset Winkeltabelle
385     P:011F 71DA00           Move            X:(R2)+,N1
                        ; Offset CAM19-15 mit ASSO
386     P:0120 57EF00           Move            X:(R7+N7),B
                        ; Berechnen von neuen Winkel
387     P:0121 200058           Add     Y0,B
388     P:0122 21E46C           Sub     X1,B    B,X0
                        ; Winkel sichern ,Verbesserung der Effizienz der CAMS
389     P:0123 0E1126           JGE     <Plus19
390     P:0124 200078           Add     Y1,B
391     P:0125 0A0020           Bset    #0,x:Test
392     P:0126 21F100 Plus19    Move            B,R1
                        ; 1. Erwarteter Hit in Lage 19- 15
393     P:0127 0D02EB           Jsr     <CAMM
                        ; Hit aus Lage 19- 15 holen (mit ASSO)
394     P:0128 20000B           Tst     B
                        ; Testen ob Suche Erfolgreich
395     P:0129 0E113B           JGE     <Hit19
                        ; Wenn Erfolgreich Sprung
396                 ;
397                 ;   1. Mal Vergebliche Suche in Lage 19 -15
398                 ;
399     P:012A 200049           Tfr     X0,B
                        ; gesuchter Winkel Lage 19- 15
400     P:012B 200068           Add     X1,B
                        ; Verbesserung der Effizienz der CAMS
401     P:012C 20007D           Cmp     Y1,B
                        ; Winkel Testen
402     P:012D 0E9130           JLT     <Ok19
                        ; Wenn Winkel Positiv Sprung
403     P:012E 20007C           Sub     Y1,B
                        ; Winkel Positiv machen
404     P:012F 0A0020           Bset    #0,X:Test
405     P:0130 21F100 Ok19      Move            B,R1
                        ; 2. Erwarteter Hit in Lage 19- 15
406     P:0131 0D02EB           Jsr     <CAMM
                        ; Hit aus Lage 19- 15 holen (mit ASSO)
407     P:0132 20000B           Tst     B
                        ; Testen ob Suche Erfolgreich
408     P:0133 0E113B           JGE     <Hit19
                        ; Wenn Erfolgreich Sprung
409                 ;
410                 ;   Vergebliche Suche in Lage 19- 15
411                 ;
412     P:0134 226F00           Move            R3,B
413     P:0135 20000B           Tst     B
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 12
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
414     P:0136 0EF0F0          JLE    <Lage20a
                        ; Neuer Wert in Lage 20 suchen
415     P:0137 045313          Lua    (R3)-,R3
                        ; Zähler erniedrigen
416     P:0138 475400          Move           Y1,X:(R4)-
                        ; Speichern des 'Nichtgefunden'
417     P:0139 200049          Tfr    X0,B
418     P:013A 0C013D          Jmp    <Loop19a
419     P:013B 575400 Hit19    Move           B,X:(R4)-
                        ; Speichern des Spurpunktes
420     P:013C 330300          Move           #Vergeb,R3
                        ; Zähler der vergeblichen Versuche
421     P:013D 000000 Loop19a  Nop
422                ;
423                ;   Lage 14 Suchen
424                ;
425     P:013E 77F400 Loop19   Move           #Tab14,N7
            0004CB          ; Offset Winkeltabelle
426     P:0140 0A000E          BClr   #14,X:Test
                        ; Testbit löschen
427     P:0141 57EF00          Move           X:(R7+N7),B
                        ; Berechnen von neuen Winkel
428     P:0142 71F458          Add    Y0,B    #CAMASS14,N1
            009E5B          ; Offset CAM14 mit ASSO
429     P:0144 21E400          Move           B,X0
                        ; Winkel sichern
430     P:0145 20006C          Sub    X1,B
                        ; Verbesserung der Effizienz der CAMS
431     P:0146 0E1149          JGE    <Plus14
432     P:0147 200078          Add    Y1,B
433     P:0148 0A0020          Bset   #0,X:Test
434     P:0149 21F100 Plus14   Move           B,R1
                        ; 1. Erwarteter Hit in Lage 14
435     P:014A 382000          Move           #Push14,N0
436     P:014B 0D02BF          Jsr    <CAM
                        ; Hit aus Lage 14 holen (mit ASSO)
437     P:014C 20000B          Tst    B
                        ; Testen ob Suche Erfolgreich
438     P:014D 0E1169          JGE    <Hit14
                        ; Wenn Erfolgreich Sprung
439                ;
440                ;   1. Mal Vergebliche Suche in Lage 14
441                ;
442     P:014E 200049          Tfr    X0,B
                        ; gesuchter Winkel Lage 14
443     P:014F 200068          Add    X1,B
                        ; Verbesserung der Effizienz der CAMS
444     P:0150 20007D          Cmp    Y1,B
                        ; Winkel Testen
445     P:0151 0E9154          JLT    <Ok14
                        ; Wenn Winkel Positiv Sprung
446     P:0152 20007C          Sub    Y1,B
                        ; Winkel Positiv machen
447     P:0153 0A0020          Bset   #0,X:Test
448     P:0154 21F100 Ok14     Move           B,R1
                        ; 2. Erwarteter Hit in Lage 14
449     P:0155 382000          Move           #Push14,N0
450     P:0156 0D02BF          Jsr    <CAM
                        ; Hit aus Lage 14 holen (mit ASSO)
451     P:0157 20000B          Tst    B
                        ; Testen ob Suche Erfolgreich
452     P:0158 0E1169          JGE    <Hit14
```

```
                        ; Wenn Erfolgreich Sprung
453                 ;
454                 ;   Vergebliche Suche in Lage 14
455                 ;
456     P:0159 226F00           Move            R3,B
457     P:015A 20000B           Tst     B
458     P:015B 0EF0F1           JLE     <Lage20
                        ; Neuer Wert in Lage 20 suchen
459     P:015C 045313           Lua     (R3)-,R3
                        ; Zähler erniedrigen
460     P:015D 470E49           Tfr     X0,B    Y1,X:P14
                        ; Vergebliche Suche
461     P:015E 0A002E           BSet    #14,X:Test
                        ;
462     P:015F 0C0189           Jmp     <Point14
463
464                 ;
465                 ;   Einsprung in Lage 14 nach vergeblicher Suche
466                 ;
467     P:0160 00008C  Lage14a  Enddo
468     P:0161 71F400  Lage14   Move            #CAM14,N1
            001E5B          ; CAM Offset ohne Asso
469     P:0163 0A00AE           Jset    #14,X:Test,Lage20
            0000F1          ; Neuer Wert in Lage 20
470     P:0165 382000           Move            #Push14,N0
471     P:0166 0D02BF           Jsr     <CAM
                        ; Hit aus Lage 14 holen (mit ASSO)
472     P:0167 20000B           Tst     B
473     P:0168 0E90F1           JLT     <Lage20
474     P:0169 570E00  Hit14    Move            B,X:P14
                        ; Speichern des Spurpunktes
475     P:016A 330300           Move            #Vergeb,R3
                        ; Zähler der vergeblichen Versuche
476     P:016B 0A000E           Bclr    #14,x:Test
477     P:016C 21E45C           Sub     Y0,B    B,X0
                        ; Sichern des Punktes,Winkeldifferenz bilden
478     P:016D 0A0080           Jclr    #0,X:Test,Null14
            00017A          ; Wurde Nullgrad Überschritten
479     P:016F 56F400           Move            #>Winkel,A
            000024          ; Ist Winkeldifferenz größer
480     P:0171 20000D           Cmp     A,B
                        ;   Maximalwert
481     P:0172 0EF176           JLE     <Null14a
482     P:0173 20007C           Sub     Y1,B
                        ; Winkel zu gross
483     P:0174 20003E           Neg     B
                        ; Maximalwinkel - Winkeldifferenz
484     P:0175 0C017A           Jmp     <Null14
485     P:0176 200036  Null14a  Neg     A
                        ; Winkel zu klein (negativ)
486     P:0177 20000D           Cmp     A,B
                        ; Maximalwinkel + Winkeldifferenz
487     P:0178 0E117A           JGE     <Null14
488     P:0179 200078           Add     Y1,B
489     P:017A 21F500  Null14   Move            B,R5
                        ;
490     P:017B 75F400           Move            #Tab14r,N5
            000421          ; Offset Radientabelle
491     P:017D 22EF00           Move            R7,B
                        ; Alter Inverser Radius
492     P:017E 56F400           Move            #Minus,A
            008000
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 14
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
493     P:0180 20000D          Cmp     A,B
494     P:0181 0E9185          JLT     <Nega14
                        ; Ist der alte Winkel negativ,
495     P:0182 56F400          Move            #Trans,A
            010000          ; dann soll es B auch sein
496     P:0184 200018          Add     A,B
497     P:0185 56ED00  Nega14  Move            X:(R5+N5),A
                        ; Inverser Radius suchen
498     P:0186 200018          Add     A,B
                        ; Mittelwert bilden
499     P:0187 20002B          Lsr     B
500     P:0188 21F700          Move            B,R7
                        ; Mittelwert sichern
501
502                ;
503                ;   Lage 13- 10 Suchen
504                ;
505     P:0189 62F400  Point14 Move            #Tab13,R2
            0005A0
506     P:018B 340D00          Move            #P13,R4
507     P:018C 060480          Do      #4,Loop13
            0001AD
508     P:018E 77DA00          Move            X:(R2)+,N7
                        ; Offset Winkeltabelle
509     P:018F 71DA00          Move            X:(R2)+,N1
                        ; Offset CAM13-10 mit ASSO
510     P:0190 57EF00          Move            X:(R7+N7),B
                        ; Berechnen von neuen Winkel
511     P:0191 200058          Add     Y0,B
512     P:0192 21E400          Move            B,X0
                        ; Winkel sichern
513     P:0193 20006C          Sub     X1,B
                        ; Verbesserung der Effizienz der CAMS
514     P:0194 0E1197          JGE     <Plus13
515     P:0195 200078          Add     Y1,B
516     P:0196 0A0020          Bset    #0,x:Test
517     P:0197 21F100  Plus13  Move            B,R1
                        ; 1. Erwarteter Hit in Lage 13 - 10
518     P:0198 0D02EB          Jsr     <CAMM
                        ; Hit aus Lage 13 - 10 holen (mit ASSO)
519     P:0199 20000B          Tst     B
                        ; Testen ob Suche Erfolgreich
520     P:019A 0E11AB          JGE     <Hit13
                        ; Wenn Erfolgreich Sprung
521                ;
522                ;   1. Mal Vergebliche Suche in Lage 13 -10
523                ;
524     P:019B 200049          Tfr     X0,B
                        ; gesuchter Winkel Lage 13- 10
525     P:019C 200068          Add     X1,B
                        ; Verbesserung der Effizienz der CAMS
526     P:019D 20007D          Cmp     Y1,B
                        ; Winkel Testen
527     P:019E 0E91A1          JLT     <Ok13
                        ; Wenn Winkel Positiv Sprung
528     P:019F 20007C          Sub     Y1,B
                        ; Winkel Positiv machen
529     P:01A0 0A0020          Bset    #0,X:Test
530     P:01A1 21F100  Ok13    Move            B,R1
                        ; 2. Erwarteter Hit in Lage 13- 10
531     P:01A2 0D02EB          Jsr     <CAMM
                        ; Hit aus Lage 13- 10 holen (mit ASSO)
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 15
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
532     P:01A3 20000B          Tst     B
                        ; Testen ob Suche Erfolgreich
533     P:01A4 0E11AB          JGE     <Hit13
                        ; Wenn Erfolgreich Sprung
534                ;
535                ;    Vergebliche Suche in Lage 13- 10
536                ;
537     P:01A5 226F00          Move            R3,B
538     P:01A6 20000B          Tst     B
539     P:01A7 0EF160          JLE     <Lage14a
                        ; Neuer Wert in Lage 16 suchen
540     P:01A8 045313          Lua     (R3)-,R3
                        ; Zähler erniedrigen
541     P:01A9 475449          Tfr     X0,B    Y1,X:(R4)-
                        ; Speichern des 'Nichtgefunden'
542     P:01AA 0C01AD          Jmp     <Loop13a
543     P:01AB 575400 Hit13    Move            B,X:(R4)-
                        ; Speichern des Spurpunktes
544     P:01AC 330300          Move            #Vergeb,R3
                        ; Zähler der vergeblichen Versuche
545     P:01AD 000000 Loop13a  Nop
546                ;
547                ;   Lage 9 Suchen
548                ;
549     P:01AE 77F400 Loop13   Move            #Tab9,N7
            000516          ; Offset Winkeltabelle
550     P:01B0 0A0009          BClr    #9,X:Test
                        ; Testbit löschen
551     P:01B1 57EF00          Move            X:(R7+N7),B
                        ; Berechnen von neuen Winkel
552     P:01B2 71F458          Add     Y0,B    #CAMASS9,N1
            00AE00          ; Offset CAM9 mit ASSO
553     P:01B4 21E400          Move            B,X0
                        ; Winkel sichern
554     P:01B5 20006C          Sub     X1,B
                        ; Verbesserung der Effizienz der CAMS
555     P:01B6 0E11B9          JGE     <Plus9
556     P:01B7 200078          Add     Y1,B
557     P:01B8 0A0020          Bset    #0,X:Test
558     P:01B9 21F100 Plus9    Move            B,R1
                        ; 1. Erwarteter Hit in Lage 9
559     P:01BA 382200          Move            #Push9,N0
560     P:01BB 0D02BF          Jsr     <CAM
                        ; Hit aus Lage 9 holen (mit ASSO)
561     P:01BC 20000B          Tst     B
                        ; Testen ob Suche Erfolgreich
562     P:01BD 0E11D9          JGE     <Hit9
                        ; Wenn Erfolgreich Sprung
563                ;
564                ;   1. Mal Vergebliche Suche in Lage 9
565                ;
566     P:01BE 200049          Tfr     X0,B
                        ; gesuchter Winkel Lage 9
567     P:01BF 200068          Add     X1,B
                        ; Verbesserung der Effizienz der CAMS
568     P:01C0 20007D          Cmp     Y1,B
                        ; Winkel Testen
569     P:01C1 0E91C4          JLT     <Ok9
                        ; Wenn Winkel Positiv Sprung
570     P:01C2 20007C          Sub     Y1,B
                        ; Winkel Positiv machen
571     P:01C3 0A0020          Bset    #0,X:Test
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 16
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
572     P:01C4 21F100  Ok9      Move            B,R1
                        ; 2. Erwarteter Hit in Lage 9
573     P:01C5 382200           Move            #Push9,N0
574     P:01C6 0D02BF           Jsr     <CAM
                        ; Hit aus Lage 9 holen (mit ASSO)
575     P:01C7 20000B           Tst     B
                        ; Testen ob Suche Erfolgreich
576     P:01C8 0E11D9           JGE     <Hit9
                        ; Wenn Erfolgreich Sprung
577                 ;
578                 ;   Vergebliche Suche in Lage 9
579                 ;
580     P:01C9 226F00           Move            R3,B
581     P:01CA 20000B           Tst     B
582     P:01CB 0EF161           JLE     <Lage14
                        ; Neuer Wert in Lage 14 suchen
583     P:01CC 045313           Lua     (R3)-,R3
                        ; Zähler erniedrigen
584     P:01CD 470949           Tfr     X0,B    Y1,X:P9
                        ; Vergebliche Suche
585     P:01CE 0A0029           BSet    #9,X:Test
586     P:01CF 0C01FA           Jmp     <Point9
587                 ;
588                 ;   Einsprung in Lage 9 nach vergeblicher Suche
589                 ;
590     P:01D0 00008C  Lage9a   Enddo
591     P:01D1 71F400  Lage9    Move            #CAM9,N1
            002E00          ; Cam Offset ohne Asso
592     P:01D3 0A00A9           Jset    #9,X:Test,Lage14
            000161          ; Neuer Wert in Lage 14
593     P:01D5 382200           Move            #Push9,N0
594     P:01D6 0D02BF           Jsr     <CAM
                        ; Hit aus Lage 9 holen (mit ASSO)
595     P:01D7 20000B           Tst     B
596     P:01D8 0E9161           JLT     <Lage14
597     P:01D9 570900  Hit9     Move            B,X:P9
                        ; Speichern des Spurpunktes
598     P:01DA 330300           Move            #Vergeb,R3
                        ; Zähler der vergeblichen Versuche
599     P:01DB 0A0009           Bclr    #9,x:Test
600     P:01DC 21E400           Move            B,X0
                        ; Sichern des Punktes
601     P:01DD 20005C           Sub     Y0,B
                        ; Winkeldifferenz bilden
602     P:01DE 0A0080           Jclr    #0,X:Test,Null9
            0001EB          ; Wurde Nullgrad Überschritten
603     P:01E0 56F400           Move            #>Winkel,A
            000024          ; Ist Winkeldifferenz größer
604     P:01E2 20000D           Cmp     A,B
                        ;   Maximalwert
605     P:01E3 0EF1E7           JLE     <Null9a
606     P:01E4 20007C           Sub     Y1,B
                        ; Winkel zu gross
607     P:01E5 20003E           Neg     B
                        ; Maximalwinkel - Winkeldifferenz
608     P:01E6 0C01EB           Jmp     <Null9
609     P:01E7 200036  Null9a   Neg     A
                        ; Winkel zu klein (negativ)
610     P:01E8 20000D           Cmp     A,B
                        ; Maximalwinkel + Winkeldifferenz
611     P:01E9 0E11EB           JGE     <Null9
612     P:01EA 200078           Add     Y1,B
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 17
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
613     P:01EB 21F500 Null9    Move            B,R5
614     P:01EC 75F400          Move            #Tab9r,N5
            000438          ; Offset Radientabelle
615     P:01EE 22EF00          Move            R7,B
                        ; Alter Inverser Radius
616     P:01EF 56F400          Move            #Minus,A
            008000
617     P:01F1 20000D          Cmp     A,B
618     P:01F2 0E91F6          JLT     <Nega9
                        ; Ist der alte Winkel negativ,
619     P:01F3 56F400          Move            #Trans,A
            010000          ; dann soll es B auch sein
620     P:01F5 200018          Add     A,B
621     P:01F6 56ED00 Nega9    Move            X:(R5+N5),A
                        ; Inverser Radius suchen
622     P:01F7 200018          Add     A,B
                        ; Mittelwert bilden
623     P:01F8 20002B          Lsr     B
624     P:01F9 21F700          Move            B,R7
                        ; Mittelwert sichern
625
626                 ;
627                 ;   Lage 8- 6 Suchen
628                 ;
629     P:01FA 62F400 Point9   Move            #Tab8,R2
            0005A8
630     P:01FC 340800          Move            #P8,R4
631     P:01FD 060380          Do      #3,Loop8
            00021D
632     P:01FF 77DA00          Move            X:(R2)+,N7
                        ; Offset Winkeltabelle
633     P:0200 71DA00          Move            X:(R2)+,N1
                        ; Offset CAM8- 6 mit ASSO
634     P:0201 57EF00          Move            X:(R7+N7),B
                        ; Berechnen von neuen Winkel
635     P:0202 200058          Add     Y0,B
636     P:0203 21E46C          Sub     X1,B    B,X0
                        ; Winkel sichern,Verbesserung der Effizienz der CAMS
637     P:0204 0E1207          JGE     <Plus8
638     P:0205 200078          Add     Y1,B
639     P:0206 0A0020          Bset    #0,x:Test
640     P:0207 21F100 Plus8    Move            B,R1
                        ; 1. Erwarteter Hit in Lage 8 - 6
641     P:0208 0D02EB          Jsr     <CAMM
                        ; Hit aus Lage 8 - 6 holen (mit ASSO)
642     P:0209 20000B          Tst     B
                        ; Testen ob Suche Erfolgreich
643     P:020A 0E121B          JGE     <Hit8
                        ; Wenn Erfolgreich Sprung
644                 ;
645                 ;   1. Mal Vergebliche Suche in Lage 8 -6
646                 ;
647     P:020B 200049          Tfr     X0,B
                        ; gesuchter Winkel Lage 8- 6
648     P:020C 200068          Add     X1,B
                        ; Verbesserung der Effizienz der CAMS
649     P:020D 20007D          Cmp     Y1,B
                        ; Winkel Testen
650     P:020E 0E9211          JLT     <Ok8
                        ; Wenn Winkel Positiv Sprung
651     P:020F 20007C          Sub     Y1,B
                        ; Winkel Positiv machen
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 18
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
652     P:0210 0A0020          Bset    #0,X:Test
653     P:0211 21F100  Ok8     Move            B,R1
                       ; 2. Erwarteter Hit in Lage 8- 6
654     P:0212 0D02EB          Jsr     <CAMM
                       ; Hit aus Lage 8- 6 holen (mit ASSO)
655     P:0213 20000B          Tst     B
                       ; Testen ob Suche Erfolgreich
656     P:0214 0E121B          JGE     <Hit8
                       ; Wenn Erfolgreich Sprung
657                ;
658                ;    Vergebliche Suche in Lage 8- 6
659                ;
660     P:0215 226F00          Move            R3,B
661     P:0216 20000B          Tst     B
662     P:0217 0EF1D0          JLE     <Lage9a
                       ; Neuer Wert in Lage 9 suchen
663     P:0218 045313          Lua     (R3)-,R3
                       ; Zähler erniedrigen
664     P:0219 475449          Tfr     X0,B    Y1,X:(R4)-
                       ; Speichern des 'Nichtgefunden'
665     P:021A 0C021D          Jmp     <Loop8a
666     P:021B 575400  Hit8    Move            B,X:(R4)-
                       ; Speichern des Spurpunktes
667     P:021C 330300          Move            #Vergeb,R3
                       ; Zähler der vergeblichen Versuche
668     P:021D 000000  Loop8a  Nop
669                ;
670                ;   Lage 5 Suchen
671                ;
672     P:021E 77F400  Loop8   Move            #Tab5,N7
            000552             ; Offset Winkeltabelle
673     P:0220 0A0005          BClr    #5,X:Test
                       ; Testbit löschen
674     P:0221 57EF00          Move            X:(R7+N7),B
                       ; Berechnen von neuen Winkel
675     P:0222 200058          Add     Y0,B
676     P:0223 71F400          Move            #CAMASS5,N1
            00BA84             ; Offset CAM5 mit ASSO
677     P:0225 21E400          Move            B,X0
                       ; Winkel sichern
678     P:0226 20006C          Sub     X1,B
                       ; Verbesserung der Effizienz der CAMS
679     P:0227 0E1229          JGE     <Plus5
680     P:0228 200078          Add     Y1,B
681     P:0229 21F100  Plus5   Move            B,R1
                       ; 1. Erwarteter Hit in Lage 5
682     P:022A 382400          Move            #Push5,N0
683     P:022B 0D02BF          Jsr     <CAM
                       ; Hit aus Lage 5 holen (mit ASSO)
684     P:022C 20000B          Tst     B
                       ; Testen ob Suche Erfolgreich
685     P:022D 0E1249          JGE     <Hit5
                       ; Wenn Erfolgreich Sprung
686                ;
687                ;   1. Mal Vergebliche Suche in Lage 5
688                ;
689     P:022E 200049          Tfr     X0,B
                       ; gesuchter Winkel Lage 5
690     P:022F 200068          Add     X1,B
                       ; Verbesserung der Effizienz der CAMS
691     P:0230 20007D          Cmp     Y1,B
                       ; Winkel Testen
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 19
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
692     P:0231 0E9233          JLT     <Ok5
                       ; Wenn Winkel Positiv Sprung
693     P:0232 20007C          Sub     Y1,B
                       ; Winkel Positiv machen
694     P:0233 21F100  Ok5     Move            B,R1
                       ; 2. Erwarteter Hit in Lage 5
695     P:0234 382400          Move            #Push5,N0
696     P:0235 0D02BF          Jsr     <CAM
                       ; Hit aus Lage 5 holen (mit ASSO)
697     P:0236 20000B          Tst     B
                       ; Testen ob Suche Erfolgreich
698     P:0237 0E1249          JGE     <Hit5
                       ; Wenn Erfolgreich Sprung
699                ;
700                ;    Vergebliche Suche in Lage 5
701                ;
702     P:0238 226F00          Move            R3,B
703     P:0239 20000B          Tst     B
704     P:023A 0EF1D1          JLE     <Lage9
                       ; Neuer Wert in Lage 9 suchen
705     P:023B 045313          Lua     (R3)-,R3
                       ; Zähler erniedrigen
706     P:023C 470500          Move            Y1,X:P5
                       ; Vergebliche Suche
707     P:023D 200049          Tfr     X0,B
708     P:023E 0A0025          BSet    #5,X:Test
                       ;
709     P:023F 0C0268          Jmp     <Point5
710                ;
711                ;    Einsprung in Lage 5 nach vergeblicher Suche
712                ;
713     P:0240 00008C  Lage5a  Enddo
714     P:0241 71F400  Lage5   Move            #CAM5,N1
            003A84          ; Cam Offset ohne Asso
715     P:0243 0A00A5          Jset    #5,X:Test,Lage9
            0001D1          ; Neuer Wert in Lage 11
716     P:0245 382400          Move            #Push5,N0
717     P:0246 0D02BF          Jsr     <CAM
                       ; Hit aus Lage 5 holen (mit ASSO)
718     P:0247 20000B          Tst     B
719     P:0248 0E91D1          JLT     <Lage9
720     P:0249 570500  Hit5    Move            B,X:P5
                       ; Speichern des Spurpunktes
721     P:024A 330300          Move            #Vergeb,R3
                       ; Zähler der vergeblichen Versuche
722     P:024B 0A0005          Bclr    #5,x:Test
723     P:024C 20005C          Sub     Y0,B
                       ; Winkeldifferenz bilden
724     P:024D 0A0080          Jclr    #0,X:Test,Null5
            00025A          ; Wurde Nullgrad Überschritten
725     P:024F 56F400          Move            #>Winkel,A
            000024          ; Ist Winkeldifferenz größer
726     P:0251 20000D          Cmp     A,B
                       ;    Maximalwert
727     P:0252 0EF256          JLE     <Null5a
728     P:0253 20007C          Sub     Y1,B
                       ; Winkel zu gross
729     P:0254 20003E          Neg     B
                       ; Maximalwinkel - Winkeldifferenz
730     P:0255 0C025A          Jmp     <Null5
731     P:0256 200036  Null5a  Neg     A
                       ; Winkel zu klein (negativ)
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 20
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
732     P:0257 20000D          Cmp     A,B
                       ; Maximalwinkel + Winkeldifferenz
733     P:0258 0E125A          JGE     <Null5
734     P:0259 200078          Add     Y1,B
735     P:025A 21F500 Null5    Move            B,R5
736     P:025B 22EF00          Move            R7,B
                       ; Alter Inverser Radius
737     P:025C 56F400          Move            #Minus,A
            008000
738     P:025E 75F40D          Cmp     A,B     #Tab5r,N5
            00044F          ; Offset Radientabelle
739     P:0260 0E9264          JLT     <Nega5
                       ; Ist der alte Winkel negativ,
740     P:0261 56F400          Move            #Trans,A
            010000          ; dann soll es B auch sein
741     P:0263 200018          Add     A,B
742     P:0264 56ED00 Nega5    Move            X:(R5+N5),A
                       ; Inverser Radius suchen
743     P:0265 200018          Add     A,B
                       ; Mittelwert bilden
744     P:0266 20002B          Lsr     B
745     P:0267 21F700          Move            B,R7
                       ; Mittelwert sichern
746                ;
747                ;   Lage 4 - 1 Suchen
748                ;
749     P:0268 62F400 Point5   Move            #Tab4,R2
            0005AE
750     P:026A 340400          Move            #P4,R4
751     P:026B 060480          Do      #4,Loop4
            00028C
752     P:026D 77DA00          Move            X:(R2)+,N7
                       ; Offset Winkeltabelle
753     P:026E 71DA00          Move            X:(R2)+,N1
                       ; Offset CAM 4 - 1 mit ASSO
754     P:026F 57EF00          Move            X:(R7+N7),B
                       ; Berechnen von neuen Winkel
755     P:0270 200058          Add     Y0,B
756     P:0271 21E46C          Sub     X1,B    B,X0
                       ; Winkel sichern,Verbesserung der Effizienz der CAMS
757     P:0272 0E1275          JGE     <Plus4
758     P:0273 200078          Add     Y1,B
759     P:0274 0A0020          Bset    #0,x:Test
760     P:0275 21F100 Plus4    Move            B,R1
                       ; 1. Erwarteter Hit in Lage 4 - 1
761     P:0276 0D02EB          Jsr     <CAMM
                       ; Hit aus Lage 4 - 1 holen (mit ASSO)
762     P:0277 20000B          Tst     B
                       ; Testen ob Suche Erfolgreich
763     P:0278 0E1289          JGE     <Hit4
                       ; Wenn Erfolgreich Sprung
764                ;
765                ;   1. Mal Vergebliche Suche in Lage 4 - 1
766                ;
767     P:0279 200049          Tfr     X0,B
                       ; gesuchter Winkel Lage 4 - 1
768     P:027A 200068          Add     X1,B
                       ; Verbesserung der Effizienz der CAMS
769     P:027B 20007D          Cmp     Y1,B
                       ; Winkel Testen
770     P:027C 0E927F          JLT     <Ok4
                       ; Wenn Winkel Positiv Sprung
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 21
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
771     P:027D 20007C          Sub     Y1,B
                       ; Winkel Positiv machen
772     P:027E 0A0020          Bset    #0,X:Test
773     P:027F 21F100  Ok4     Move            B,R1
                       ; 2. Erwarteter Hit in Lage 4 - 1
774     P:0280 0D02EB          Jsr     <CAMM
                       ; Hit aus Lage 4 - 1 holen (mit ASSO)
775     P:0281 20000B          Tst     B
                       ; Testen ob Suche Erfolgreich
776     P:0282 0E1289          JGE     <Hit4
                       ; Wenn Erfolgreich Sprung
777                ;
778                ;    Vergebliche Suche in Lage 4 - 1
779                ;
780     P:0283 226F00          Move            R3,B
781     P:0284 20000B          Tst     B
782     P:0285 0EF240          JLE     <Lage5a
                       ; Neuer Wert in Lage 5 suchen
783     P:0286 045313          Lua     (R3)-,R3
                       ; Zähler erniedrigen
784     P:0287 475449          Tfr     X0,B    Y1,X:(R4)-
                       ; Speichern des 'Nichtgefunden'
785     P:0288 0C028C          Jmp     <Loop4a
786     P:0289 000000  Hit4    Nop
787     P:028A 575400          Move            B,X:(R4)-
                       ; Speichern des Spurpunktes
788     P:028B 330300          Move            #Vergeb,R3
                       ; Zähler der vergeblichen Versuche
789     P:028C 000000  Loop4a  Nop
790                ;
791                ;    Spur sichern
792                ;
793     P:028D 321800  Loop4   Move            #P24,R2
794     P:028E 0618A0          Rep     #24
795     P:028F 08D2AB          Movep           X:(R2)-,X:<<$ffeb
                       ;punkte
796     P:0290 64A600          Move            X:Trkcnt,R4
797     P:0291 08D72B          Movep           R7,X:<<$ffeb
                       ;radius  sichern
798     P:0292 045414          Lua     (R4)-,R4
799     P:0293 331700          Move            #P23,R3
800     P:0294 228F00          Move            R4,B
                       ; pruefen, ob schon genug
801     P:0295 20000B          Tst     B
                       ; Spuren gefunden worden
802     P:0296 0AF0AA          JEQ     Fin
            0002B5         ;sind
803     P:0298 642600          Move            R4,X:Trkcnt
804     P:0299 64F400          Move            #Tabx,R4
            0005B6
805     P:029B 20001B          Clr     B
806     P:029C 044FBF          Move            LC,B
807     P:029D 20002F          Ror     B
808     P:029E 0AF0A8          JCS     Spiegel
            0002A5
809     P:02A0 044FBF          Move            LC,B
810     P:02A1 46F400          Move            #>1,Y0
            000001
811     P:02A3 20005C          Sub     Y0,B
812     P:02A4 04CFBF          Move            B,LC
813     P:02A5 56F400  Spiegel Move            #>1,A
            000001
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 22
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
Programm

```
814     P:02A7 200036          Neg     A
815     P:02A8 061780          Do      #23,End24
            0002B0
816     P:02AA 72D300           Move            X:(R3)-,N2
817     P:02AB 62DC00           Move            X:(R4)+,R2
818     P:02AC 000000          Nop
819     P:02AD 5E6A00           Move                    A,Y:(R2+N2)
820     P:02AE 62DC00           Move            X:(R4)+,R2
821     P:02AF 000000          Nop
822     P:02B0 5E6A00           Move                    A,Y:(R2+N2)
823     P:02B1 000000  End24    Nop
824     P:02B2 000000  Loop24   Nop
825     P:02B3 0AF080           Jmp     FIN
            0002BB
826     P:02B5 00008C  Fin      Enddo
827     P:02B6 000000  Fina     Nop
828     P:02B7 000000          Nop
829     P:02B8 08F4AB  Ende     Movep           #Ok,X:<<$ffeb
            FFFFFF          ; Nehme Spur
830     P:02BA 000087           Stop
831     P:02BB 0C02B6  FIN      Jmp     Fina
                        ; Dieser Sprung ist nur für die Simulat.
832     P:02BC 08F4AB           Movep           #No,X:<<$ffeb
            0AFFE0          ; Verwerfe Spur
833     P:02BE 000087           Stop
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 23
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
CAM Simulationsroutine

```
836
                        ;
837
                        ;  Subroutine Cam- Simulation
838
                        ;
839     P:02BF 232E00  CAM      Move            N1,A
840     P:02C0 57F400           Move            #$8000,B
            008000
841     P:02C2 711900           Move            N1,X:Push1
842     P:02C3 611A00           Move            R1,X:Push2
843     P:02C4 441B00           Move            X0,X:Push3
844     P:02C5 20000D           Cmp     A,B
845     P:02C6 0EF2CF           JLE     <ASSO
846     P:02C7 231100           Move            N0,R1
847     P:02C8 000000           Nop
848     P:02C9 71D900           Move            X:(R1)+,N1
849     P:02CA 61E100           Move            X:(R1),R1
850     P:02CB 222E00           Move            R1,A
851     P:02CC 200003           Tst     A
852     P:02CD 0EF2DF           JLE     <Subr
853     P:02CE 0C02D5           Jmp     <DoLoop
854     P:02CF 222F00  ASSO     Move            R1,B
855     P:02D0 44F400           Move            #$fffffe,X0
            FFFFFE
856     P:02D2 232E4E           And     X0,B    N1,A
857     P:02D3 310418           Add     A,B     #4,R1
858     P:02D4 21F900           Move            B,N1
859     P:02D5 06D100  DoLoop   Do      R1,Subr
            0002DE
860     P:02D7 045111           Lua     (R1)-,R1
861     P:02D8 5FE900           Move                    Y:(R1+N1),B
862     P:02D9 20000B           Tst     B
863     P:02DA 0E92DE           JLT     <Subra
864     P:02DB 00008C           Enddo
865     P:02DC 0C02E2           Jmp     <Found
866     P:02DD 000000           Nop
867     P:02DE 000000  Subra    Nop
868     P:02DF 000000  Subr     Nop
869     P:02E0 57F400           Move            #$ffffff,B
            FFFFFF
870     P:02E2 222E00  Found    Move            R1,A
871     P:02E3 231100           Move            N0,R1
872     P:02E4 000000           Nop
873     P:02E5 715900           Move            N1,X:(R1)+
874     P:02E6 566100           Move            A,X:(R1)
875     P:02E7 719900           Move            X:Push1,N1
876     P:02E8 619A00           Move            X:Push2,R1
877     P:02E9 449B00           Move            X:Push3,X0
878     P:02EA 00000C           Rts
879                 ;
880                 ;  Cam Simulation mit ASSO
881                 ;
882     P:02EB 711900  CAMM     Move            N1,X:Push1
883     P:02EC 611A00           Move            R1,X:Push2
884     P:02ED 441B00           Move            X0,X:Push3
885     P:02EE 222F00           Move            R1,B
886     P:02EF 44F400           Move            #$fffffe,X0
            FFFFFE
887     P:02F1 232E4E           And     X0,B    N1,A
888     P:02F2 310418           Add     A,B     #4,R1
889     P:02F3 21F900           Move            B,N1
```

Motorola DSP56000 Macro Cross Assembler Version 1.00 02-02-88 15:45:17 version9.asm
Seite 24
Stufe 3 Trigger Algorithmus (MONIKA) auf DSP56000 von Motorola mit CAM- Simulation
CAM Simulationsroutine

```
890     P:02F4 06D100          Do      R1,Subr1
            0002FD
891     P:02F6 045111          Lua     (R1)-,R1
892     P:02F7 5FE900          Move                    Y:(R1+N1),B
893     P:02F8 20000B          Tst     B
894     P:02F9 0E92FD          JLT     <Subr1a
895     P:02FA 00008C          Enddo
896     P:02FB 0C0301          Jmp     <Found1
897     P:02FC 000000          Nop
898     P:02FD 000000  Subr1a  Nop
899     P:02FE 000000  Subr1   Nop
900     P:02FF 57F400          Move            #$fffff,B
            FFFFFF
901     P:0301 719900  Found1  Move             X:Push1,N1
902     P:0302 619A00          Move            X:Push2,R1
903     P:0303 449B00          Move            X:Push3,X0
904     P:0304 00000C          Rts
905     P:E000                 org     p:$e000
906     P:E000 0C0040          Jmp     Begin
                       ; des kleinen Bug im Simulator wegen
```

VI FORTRAN Programme

VI.1. Simulation des Maskenprozessors

Dies ist das verwendete FORTRAN77 Programm zur Simulation eines Triggers mit einem Maskenprozessor. Es verwendet GEP77 Routinen zur Darstellung des Ergebnisses. Die verwendete JCL ist ebenfalls aufgeführt.

```
//DARBOLUO JOB '00010221,F36WEN',CLASS=L,TIME=(10,00),
// MSGLEVEL=(0,0),
// NOTIFY=F36WEN
// EXEC JFORTCLG,CPRM='NOSOURCE,OPTIMIZE(3),NOPRINT'
//*
//C.SYSIN DD *
      PROGRAM GATTER
      INTEGER      VERT,HOR,CELL
      REAL         RADIUS
      INTEGER      TREFFE,TREFF,WINKEL,KAMMER,EVENT,NUMMER
      LOGICAL       ANFANG
      DIMENSION   VERT(25,50),HOR(25,50),CELL(25,50),RADIUS(24)
      COMMON /GEP/ TREFFE,  TREFF,ANFANG
      COMMON/DATEN/WINKEL,KAMMER,EVENT,NUMMER
      COMMON/AREA/VERT,HOR,CELL,RADIUS
C
C    Hauptprogramm
C
C     CALL GEPIC
1     CALL INITT
      ANFANG=.FALSE.
      VORZ=1
      CALL EINLES(*2)
      GOTO 1
C2     CALL GEPW
 2    STOP
      END

C
C  ********************************************************************
C
      SUBROUTINE INITT
      INTEGER      TREFF,TREFFE
      INTEGER      VERT,HOR,CELL
      LOGICAL      ANFANG
      REAL         RADIUS
      COMMON/AREA/VERT(25,50),HOR(25,50),CELL(25,50),RADIUS(24)
      COMMON /GEP/ TREFFE, TREFF,ANFANG
C
C   Dieses Unterprogramm initialisiert die Arrays
C   VERT,HOR,CELL
C
      DO 2 I=1,25
         DO 1 II=1,50
         VERT(I,II)=0
         HOR(I,II)=0
         CELL(I,II)=0
 1       CONTINUE
 2    CONTINUE
C
C   Radien der Kammern
C
      RADIUS(1)=21.83
      RADIUS(2)=22.62
      RADIUS(3)=23.44
      RADIUS(4)=24.26
```

```
      RADIUS(5)=25.11
      RADIUS(6)=25.96
      RADIUS(7)=26.83
      RADIUS(8)=27.70
      RADIUS(9)=28.59
      RADIUS(10)=29.48
      RADIUS(11)=30.39
      RADIUS(12)=31.29
      RADIUS(13)=32.21
      RADIUS(14)=33.13
      RADIUS(15)=34.06
      RADIUS(16)=34.98
      RADIUS(17)=35.93
      RADIUS(18)=36.86
      RADIUS(19)=37.81
      RADIUS(20)=38.75
      RADIUS(21)=39.71
      RADIUS(22)=40.66
      RADIUS(23)=41.62
      RADIUS(24)=42.57
      RETURN
      END
C
C     ************************************************************
C
      SUBROUTINE SETZEN(WINKEL,KAMMER,HOCH,SEITE)
      INTEGER       VERT,HOR,CELL,KAMMER,KAMMR1,WINKEL,WINKL1
      REAL          RADIUS
      INTEGER       KAMMR,WINKL
      INTEGER       HOCH,SEITE,MOD,INT
      COMMON/AREA/VERT(25,50),HOR(25,50),CELL(25,50),RADIUS(24)
C
C   Diese Procedure soll die HITS eintragen in das Gitter.
C   Als Parameter werden uebergeben die X-Koordinate, die
C   Y-Koordinate, die Lage, und zwei Parameter fuer das
C   einrichten verschiedener Masken.
C
      KAMMR=KAMMER+HOCH
      WINKL=WINKEL+SEITE
 121  FORMAT(4I12)
      IF ((KAMMR.LE.0).OR. (KAMMR.GE.25)) GOTO 1
      IF (WINKL.LT.0)   WINKL = WINKL+(INT(ABS(WINKL)/800)+1)*800
      IF (WINKL.GT.799) WINKL = WINKL- INT(WINKL/800)*800
      HOR(KAMMR,INT(WINKL/16)+1) =
     *          IBSET(HOR(KAMMR,INT(WINKL/16)+1),MOD(WINKL,16))
      VERT(KAMMR,INT(WINKL/16)+1) =
     *          IBSET(VERT(KAMMR,INT(WINKL/16)+1),MOD(WINKL,16))
      WINKL1= WINKL-1
      KAMMR1 = KAMMR+1
      IF (WINKL1.LT.0) WINKL1=799
      HOR(KAMMR,INT(WINKL1/16)+1) =
     *          IBSET(HOR(KAMMR,INT(WINKL1/16)+1),MOD(WINKL1,16))
      IF (KAMMR1.GT.25) GOTO 1
      VERT(KAMMR1,INT(WINKL/16)+1) =
     *          IBSET(VERT(KAMMR1,INT(WINKL/16)+1),MOD(WINKL,16))
 1    RETURN
      END
C
C     ************************************************************
C
      SUBROUTINE EINLES(*)
      INTEGER       WINKEL,KAMMER,EVENT,NUMMER
      LOGICAL        ANFANG
      COMMON /DATEN/ WINKEL,KAMMER,EVENT,NUMMER
      DATA TWOPI,PI/6.283185307,3.1415926535/

C
```

```
C   Diese Procedure liesst die Daten vom File und ruft dann
C   die Routine zur Erzeugung von den Masken in dem Gitter auf.
C   Der Parameter NUMMER sagt welches EVENT schon gelesen worden ist
C
C   RETURN  = ein Erreignis ist vollstaendig
C   RETURN1 = kein weiteres Erreignis
C
      NUMMER=EVENT
C      CALL DEKB(NUMMER)
 1    IF(ANFANG) GOTO 33
 11   READ(12,END=6,ERR=5) INUM,JEVENT,IPAR,ICH,JCH,AX,AY,AZ
      IF((INUM.NE.29).OR.(ICH.NE.1)) GOTO11
      GOTO13
 13   ANFANG= .TRUE.
      PHI=ATAN(AY/AX)
      IF (AX.LT..0) PHI=PHI+PI
      IF ((AX.GT..0).AND.(AY.LT..0)) PHI=PHI+TWOPI
      PHI=PHI/TWOPI*800.
      WINKEL=INT(PHI)
      KAMMER=JCH
      EVENT=JEVENT
 2    NUMMER=EVENT
C      CALL DEKB(NUMMER)
      WRITE(6,21) EVENT
 21   FORMAT(' EVENT ',I3,' ANFANG!')
      GOTO  4
 3    READ(12,END=6,ERR=5) INUM,JEVENT,IPAR,ICH,JCH,AX,AY,AZ
      IF((INUM.NE.29).OR.(ICH.NE.1)) GOTO3
      GOTO 32
 32   PHI=ATAN(AY/AX)
      IF (AX.LT..0) PHI=PHI+PI
      IF ((AX.GT..0).AND.(AY.LT..0)) PHI=PHI+TWOPI
      PHI=PHI/TWOPI*800.
      WINKEL=INT(PHI)
      KAMMER=INT(JCH)
      EVENT=INT(JEVENT)
33    IF (NUMMER.NE.EVENT) GOTO 100
C
C    Hier stehen KAMMER und WINKEL zur Verfuegung
C
 4         CALL MASKE(KAMMER,WINKEL,+1)
           GOTO 3
 100  CALL FINDE
      RETURN
 6    CALL FINDE
 5    RETURN1
      END
C
C     ************************************************************
C
      SUBROUTINE DRAW(KAMMER,WINKEL)
      INTEGER      KAMMER,WINKEL,VERT,HOR,CELL,WINKL,KAMMR,NUMMER,EVENT
      REAL         RADIUS
      COMMON/AREA/VERT(25,50),HOR(25,50),CELL(25,50),RADIUS(24)
      COMMON /DATEN/ WINKL,KAMMR,EVENT,NUMMER
      DATA FAKTOR/0.007853981634/
C
C   Diese Subroutine traegt die HITS in eine GEP Datei
C
C      CALL GENU(1,-2,RADIUS(KAMMER)*COS(WINKEL*FAKTOR),
C    *               RADIUS(KAMMER)*SIN(WINKEL*FAKTOR))
      WRITE(10)KAMMER,WINKEL,NUMMER
      RETURN
      END
C
C     ************************************************************
C
```

```
      SUBROUTINE MASKE(KAMMER,WINKEL,VORZ)
      INTEGER      KAMMER,WINKEL
      INTEGER      VORZ,VOR
C
C   Diese Subroutine legt die Maske fest und laesst Sie Setzen
C
      CALL DRAW(KAMMER,WINKEL)
      IF (VORZ.GE.0) VOR=1
      IF (VORZ.LT.0) VOR=-1
      DO 2 I=0,3
         DO 1 II=0,I,1
             CALL SETZEN(WINKEL,KAMMER,I,INT(VOR*II))
  1          CONTINUE
  2      CONTINUE
      RETURN
      END
C
C     ********************************************************
C
      SUBROUTINE FINDE
      INTEGER      VERT,HOR,CELL,TEST,PRIOR,K,KK,KKK,NUMMR
      REAL         RADIUS
      INTEGER      WINKEL,KAMMER,EVENT,HILF,WORT,I,TREFF,TREFFE
      LOGICAL      HIT,ANFANG
      CHARACTER*6  ASCII
      COMMON/AREA/VERT(25,50),HOR(25,50),CELL(25,50),RADIUS(24)
      DIMENSION TEST(50)
      COMMON /GEP/ TREFFE,  TREFF,ANFANG
      COMMON /ASCII/ASCII
      COMMON /DATEN/ WINKEL,KAMMER,EVENT,NUMMR
      DATA FAKTOR/0.007853981634/
C
C    Diese SUBROUTINE soll die Steuerung des Gitters zur
C    Spursuche simulieren.
C
      TREFFE=TREFF
      DO 1 I=1,50
        CELL(1,I)=-1
  1     CONTINUE
      DO 2 I=1,24,1
        CALL GITTER(I,.TRUE.,.FALSE.,*8)
  2     CONTINUE
      DO 3 I=1,50
        TEST(I)   = CELL(25,I)
  3     CONTINUE
      WORT=1
  5   DO 4 I=1,50
        DO 55 II=1,25
           CELL(II,I)=0
  55       CONTINUE
  4     CONTINUE
  44  CELL(25,WORT)=PRIOR(TEST(WORT))
      IF (CELL(25,WORT).EQ.0) GOTO 777
  47    FORMAT(2I20)
      CALL GITTER(24,.TRUE.,.TRUE.,*8)
      DO 45 I=1,50
          TEST(I)=IAND(TEST(I),NOT(CELL(25,I)))
  45      CONTINUE
      DO 6 I=24,1,-1
         CALL GITTER(I,.FALSE.,.TRUE.,*8)
  6      CONTINUE
      HIT=.FALSE.
      DO 7 II=1,50
         IF (CELL(1,II).NE.0) HIT=.TRUE.
  7      CONTINUE
      IF (.NOT.HIT) GOTO 75
      TREFF=TREFF+1
```

```
      WRITE(6,77) TREFF-TREFFE,TREFF
 77   FORMAT ( ' Bis jetzt ',I3,' Treffer in Event,'
     *        ,'Insgesamt ',I3,' Treffer!')
 75   DO 78 K=2,25
        DO 79 KK=1,50
          DO 74 KKK=0,15
            HILF=KK*16-16+KKK
C           IF (BTEST(CELL(K,KK),KKK)) CALL GENU(2,3,
C    *                        RADIUS(K-1)*COS(FAKTOR*HILF),
C    *                        RADIUS(K-1)*SIN(FAKTOR*HILF))
            IF (BTEST(CELL(K,KK),KKK)) WRITE(10)
     *                 KAMMER,WINKEL,NUMMR
 74         CONTINUE
 79       CONTINUE
 78     CONTINUE
      GOTO 5
 777  WORT=WORT+1
      IF (WORT.LT.51) GOTO 44
      GOTO 5
 8    WRITE(6,9)
 9    FORMAT('     ERROR IN GITTER    ')
C      CALL HIST(3,TREFF-TREFFE)
      RETURN
      END
C
C     ***********************************************************
C
      SUBROUTINE KONVER(VAR)
      CHARACTER*6 ASCII
      INTEGER      INPUT,DUMMY,WERT,VAR
      COMMON /ASCII/ASCII
C
C    Diese Routine Konvertiert Integer auf ASCII
C    Zeichenfolge
C
      WERT=10000
      ASCII=' '
      WERT=VAR
1     DUMMY=INT(INPUT/WERT)
      ASCII=ASCII//CHAR(DUMMY+48)
      INPUT=INPUT-DUMMY*WERT
      WERT=INT(WERT/10)
      IF (WERT.GT.0) GOTO 1
      ASCII=ASCII//CHAR(WERT+48)
      END
C
C     ***********************************************************
C
      SUBROUTINE GITTER(POINT,UP,DOWN,*)
      INTEGER      VERT,HOR,CELL
      REAL         RADIUS
      INTEGER      POINT,I
      LOGICAL      UP,DOWN,TEST,BTEST
      COMMON/AREA/VERT(25,50),HOR(25,50),CELL(25,50),RADIUS(24)
C
C   Diese Subroutine simuliert die Gatter des
C   Spezialprozessors von Darbo veraendert fuer
C   die andere Situation von H1
C
C   Bei den Arrays VERT,HOR,CELL (A,B) ist A die
C   Lage und B das Wort in der Lage.
C   Es gibt pro Lage 50 Woerter.
C   CELL enth lt die logische Information der
C   Knoten, HOR die Information ob horizontale
C   Verbindungen bestehen und VERT die Information
C   ob vertikale Verbindungen bestehen.
C
```

```
C   Return1 = Point ist zu klein oder zu gross
C   Return  = Alles in Ordnung
C
      IF ((POINT.GT.24).OR.(POINT.LT.1)) WRITE(6,2) POINT
 2    FORMAT(' POINT= ',1I9)
      IF ((POINT.GT.24).OR.(POINT.LT.1)) RETURN1
      I=POINT
      IF (DOWN) GOTO 140
C
C   Aufwaerts arbeiten
C
 1    DO 10 II=1,50,1
         CELL(I+1,II)=IAND(VERT(I,II),CELL(I,II))
 10   CONTINUE
      DO 40 II=1,50,1
         DO 20 III=0,14,1
         TEST=(BTEST(HOR(I,II),III).AND.(BTEST(CELL(I+1,II),III)))
         IF (TEST) CELL(I+1,II)=IBSET(CELL(I+1,II),III+1)
 20      CONTINUE
      IF (II.EQ.50) GOTO 40
      TEST=(BTEST(HOR(I,II),15).AND.(BTEST(CELL(I+1,II),15)))
      IF (TEST)    CELL(I+1,II+1)=IBSET(CELL((I+1),II+1),0)
 40   CONTINUE
      DO 100 II=50,1,-1
         DO 80 III=15,1,-1
         TEST=(BTEST(HOR(I,II),(III-1)).AND.(BTEST(CELL(I+1,II),III)))
         IF (TEST) CELL(I+1,II)=IBSET(CELL(I+1,II),III-1)
 80      CONTINUE
      IF (II.EQ.1) GOTO 100
         TEST=(BTEST(HOR(I,II-1),15).AND.(BTEST(CELL(I+1,II),0)))
         IF (TEST) CELL(I+1,II-1)=IBSET(CELL((I+1),II-1),15)
 100  CONTINUE
      RETURN
C
C   Abwaerts arbeiten
C
 140     DO 170 II=1,50,1
         DO 150 III=0,14,1
         TEST=(BTEST(HOR(I,II),III).AND.(BTEST(CELL((I+1),II),III)))
         IF (TEST) CELL((I+1),II)=IBSET(CELL((I+1),II),III+1)
 150     CONTINUE
      IF (II.EQ.50) GOTO 170
      TEST=(BTEST(HOR(I,II),15).AND.(BTEST(CELL(I+1,II),15)))
      IF (TEST)    CELL(I+1,II+1)=IBSET(CELL((I+1),II+1),0)
 170  CONTINUE
      DO 230 II=50,1,-1
         DO 210 III=15,1,-1
         TEST=(BTEST(HOR(I,II),(III-1)).AND.(BTEST(CELL(I+1,II),III)))
         IF (TEST) CELL(I+1,II)=IBSET(CELL(I+1,II),III-1)
 210     CONTINUE
      IF (II.EQ.1) GOTO 230
         TEST=(BTEST(HOR(I,II-1),15).AND.(BTEST(CELL(I+1,II),0)))
         IF (TEST) CELL(I+1,II-1)=IBSET(CELL((I+1),II-1),15)
 230  CONTINUE
      IF (UP) GOTO 300
      DO 270 II=1,50,1
         CELL(I,II)=IAND(VERT(I,II),CELL(I+1,II))
 270     CONTINUE
          RETURN
C
C    Falls up und down ist, werden noch die
C    folgende Zeilen ausgefuehrt
C
 300  DO 310 II=1,50,1
          CELL(I,II)=CELL(I+1,II)
 310      CONTINUE
         I=POINT-1
```

```
         IF (I.LE.1) RETURN 1
         DO 330 II=1,50,1
         DO 320 III=0,15,1
         TEST=(BTEST(HOR(I,II),III).AND.(BTEST(CELL((I+1),II),III)))
         IF (TEST) CELL((I+1),II)=IBSET(CELL((I+1),II),III+1)
 320     CONTINUE
      IF (II.EQ.50) GOTO 330
      TEST=(BTEST(HOR(I,II),15).AND.(BTEST(CELL(I+1,II),15)))
      IF (TEST)    CELL(I+1,II+1)=IBSET(CELL((I+1),II+1),0)
 330  CONTINUE
      DO 350 II=50,1,-1
         DO 340 III=15,1,-1
         TEST=(BTEST(HOR(I,II),(III-1)).AND.(BTEST(CELL(I+1,II),III)))
         IF (TEST) CELL(I+1,II)=IBSET(CELL(I+1,II),III-1)
 340     CONTINUE
         IF (II.EQ.1) GOTO 350
         TEST=(BTEST(HOR(I,II-1),15).AND.(BTEST(CELL(I+1,II),0)))
         IF (TEST) CELL(I+1,II-1)=IBSET(CELL((I+1),II-1),15)
 350  CONTINUE
      I=POINT
      DO 360 II=1,50
      CELL(I+1,II)=IOR(CELL(I,II),CELL(I+1,II))
 360  CONTINUE
      DO 380 II=1,I
         DO 370 III=1,50
         CELL(II,III)=0
 370     CONTINUE
 380  CONTINUE
      RETURN
      END
C
C*************************************************************
C
C
C   FUNCTION Definition
C
C
C  EIN TEIL DES PRIORITYENCODER
C
      INTEGER    FUNCTION PRIOR(LWORT)
           LOGICAL      BTEST
           INTEGER      I
           DO 1 I=15,0,-1
              PRIOR=IBSET(0,I)
              IF (BTEST(LWORT,I)) GOTO 2
   1          CONTINUE
           PRIOR=0
   2       END
//G.SYSIN DD *
NOPRINT
DST 1 'SIMULIERTE HITS
DST 2 'GEFUNDENE HITS
DST 3,1 'ANZAHL SPUREN PRO EVENT
BINS 1,-0.5,-40,5,41
ENDQ
//G.FT12F001 DD DSN='F36HJB.GEANT.LU2',DISP=SHR
//G.FT10F001 DD DSN='F36WEN.GEANT.LU2',DISP=SHR
/*   ENDE
```

VI.2. Programm für Kreisgleichungsalgorithmus

Dies ist das verwendete FORTRAN77 Programm zur Simulation der "*Spursuche unter Verwendung der Kreisgleichung*":

```
//* 13/03/87 703311637  MEMBER NAME  EFFIZ    (S)          JCL
//EFFITZ0 JOB '00010221,F36WEN',CLASS=E,TIME=(0,14),
// MSGLEVEL=(1,1),
// NOTIFY=F36WEN
//    EXEC VFORTCLG
//*
//C.SYSIN DD *
C   16/02/87 703122000  MEMBER NAME  TEST     (MONIKA)      FORTRAN77
      BLOCK DATA
      COMMON/CONST/MIN,AUSGAB,INDSET,IND,TWOPI,MAXVER,VERGEB
      COMMON/CAMM/MASKE,WEITE
      COMMON/REGIST/REG,SPURDR,NSPUR
      COMMON/TREF/HIT
      COMMON/VAR/SPUR,PUNKT,TREFFE,R,P100
      COMMON/MASK/MASK
      COMMON/BINNS/BINN
      COMMON/START/ANFANG,ANZAHL
      LOGICAL    AUSGAB,SPURDR,ANFANG
      INTEGER*2  MASKE,WEITE,MASK
      INTEGER    INDSET,IND,MAXVER,VERGEB
     *          ,SPUR,TREFFE,NSPUR,BINN
      REAL       TWOPI,MIN,REG,R,P100,PUNKT
      DIMENSION MASKE(24),WEITE(24),REG(15),HIT(50,24),PUNKT(24)
     *,TREFFE(24),SPUR(24)
C
C  AUSGAB IST EINE STEUERVARIABLE FUER DRUCKFUNKTIONENDER ZWISCHENWERTE
C  SPURDR IST EINE STEUERVARIABLE FUER DRUCKFUNKTION AM ENDE DER SUCHE
C  MIN IST DER KLEINSTE RADIUS ENTSPRICHT 277CM BEI 1GEV
C  TWOPI  2* PI
C  INDSET  MAXIMALE ZAHL DER VERSUCHE OHNE TREFFER IN LAGE 22
C  MAXVER  MAXIMALE ZAHL DER LAGEN OHNE GEFUNDEN TREFFER
C
      DATA MIN,INDSET,TWOPI/2.5,2,6.283185307/
      DATA AUSGAB/.FALSE./,SPURDR/.FALSE./,ANFANG/.FALSE./
      DATA MAXVER/5/
C
C WENN MAN VON EINER ORTSAUFLOESUNG VON .5 MM AUSGEHT,
C SO HAT MAN MAXIMAL 13363 BINS
C 2 * PI ENTSPRICHT ALSO 13363 !!
C
C FESTLEGUNG DER MASKEN FUER DIE CAM-ASSOSIATION
C
C     ZAHL DER      |       |
C   MASKIERTEN BITS | MASKE | WEITE
C ---------------------------------
C       16  BIT     | Z0000 | 65535
C       15  BIT     | Z8000 | 32767
C       14  BIT     | ZC000 | 16383
C       13  BIT     | ZE000 |  8191
C       12  BIT     | ZF000 |  4095
C       11  BIT     | ZF800 |  2047
C       10  BIT     | ZFC00 |  1023
C        9  BIT     | ZFE00 |   501
C        8  BIT     | ZFF00 |   255
C        7  BIT     | ZFF80 |   127
C        6  BIT     | ZFFC0 |    63
C        5  BIT     | ZFFE0 |    31
```

```
C        4  BIT      | ZFFF0 |    15
C        3  BIT      | ZFFF8 |     7
C        2  BIT      | ZFFFC |     3
C        1  BIT      | ZFFFE |     1
C        0  BIT      | ZFFFF |     0
C
C  1. DATENSATZ
C      DATA MASKE/ZFE00,ZFE00,ZFE00,ZFE00,ZFE00,ZFF00,ZFF00,ZFF00,ZFF00,
C     *           ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,
C     *           ZFF00,ZFF00,ZFF00,ZFF00,ZFFC0,ZFFFF/
C      DATA WEITE/501  ,511  ,511  ,511  ,511  ,255  ,255  ,255  ,255  ,
C     *           255  ,255  ,255  ,255  ,255  ,255  ,255  ,255  ,255  ,
C     *           255  ,255  ,255  ,255  ,1023 ,65534/
C DAS ENTSPRICHT/9BIT ,9BIT ,9BIT ,9BIT ,9BIT ,8BIT ,8BIT ,8BIT ,8BIT ,
C                 8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,
C                 8BIT ,8BIT ,8BIT,8BIT,10BIT,16BIT/
C 2. DATENSATZ
      DATA MASKE/ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFFC0,ZFFC0,ZFFC0,ZFFC0,
     *           ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,
     *           ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFFF/
      DATA WEITE/127  ,127  ,127  ,127  ,127  ,63   ,63   ,63   ,63   ,
     *           63   ,63   ,63   ,63   ,63   ,63   ,63   ,63   ,63   ,
     *           63   ,63   ,63   ,63   ,63   ,-1   /
C DAS ENTSPRICHT/7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,6BIT ,6BIT ,6BIT ,6BIT ,
C                 6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,
C                 6BIT ,6BIT ,6BIT,6BIT,6BIT,16BIT/
C 3. MASKENSATZ
C
C      DATA MASKE/ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFFC0,ZFFC0,ZFFC0,ZFFC0,
C     *           ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,
C     *           ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFF80,ZFFFF/
C      DATA WEITE/127  ,127  ,127  ,127  ,127  ,63   ,63   ,63   ,63   ,
C     *           63   ,63   ,63   ,63   ,63   ,63   ,63   ,63   ,63   ,
C     *           63   ,63   ,63   ,63   ,127  ,-1   /
C DAS ENTSPRICHT/7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,6BIT ,6BIT ,6BIT ,6BIT ,
C                 6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,
C                 6BIT ,6BIT ,6BIT,6BIT,7BIT,16BIT/
C 4. MASKENSATZ
C
C      DATA MASKE/ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFFC0,ZFFC0,ZFFC0,ZFFC0,
C     *           ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,
C     *           ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFE00,ZFFFF/
C      DATA WEITE/127  ,127  ,127  ,127  ,127  ,63   ,63   ,63   ,63   ,
C     *           63   ,63   ,63   ,63   ,63   ,63   ,63   ,63   ,63   ,
C     *           63   ,63   ,63   ,63   ,501  ,-1   /
C DAS ENTSPRICHT/7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,6BIT ,6BIT ,6BIT ,6BIT ,
C                 6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,
C                 6BIT ,6BIT ,6BIT,6BIT,9BIT,16BIT/
C 5. MASKENSATZ
C
C      DATA MASKE/ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFFC0,ZFFC0,ZFFC0,ZFFC0,
C     *           ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFFC0,
C     *           ZFFC0,ZFFC0,ZFFC0,ZFFC0,ZFF00,ZFFFF/
C      DATA WEITE/127  ,127  ,127  ,127  ,127  ,63   ,63   ,63   ,63   ,
C     *           63   ,63   ,63   ,63   ,63   ,63   ,63   ,63   ,63   ,
C     *           63   ,63   ,63   ,63   ,255  ,-1   /
C DAS ENTSPRICHT/7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,6BIT ,6BIT ,6BIT ,6BIT ,
C                 6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,6BIT ,
C                 6BIT ,6BIT ,6BIT,6BIT,8BIT,16BIT/
C 6. MASKENSATZ
C
C      DATA MASKE/ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF80,ZFF80,ZFF80,ZFF80,
C     *           ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,
C     *           ZFF80,ZFF80,ZFF80,ZFF80,ZFF00,ZFFFF/
C      DATA WEITE/255  ,255  ,255  ,255  ,255  ,127  ,127  ,127  ,127  ,
C     *           127  ,127  ,127  ,127  ,127  ,127  ,127  ,127  ,127  ,
C     *           127  ,127  ,127  ,127  ,255  ,-1   /
```

```
C DAS ENTSPRICHT/8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,7BIT ,7BIT ,7BIT ,7BIT ,
C                7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,
C                7BIT ,7BIT ,7BIT,7BIT,8BIT,17BIT/
C 7. MASKENSATZ
C
C     DATA MASKE/ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF80,ZFF80,ZFF80,ZFF80,
C    *           ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,ZFF80,
C    *           ZFF80,ZFF80,ZFF80,ZFF80,ZFE00,ZFFFF/
C     DATA WEITE/255  ,255  ,255  ,255  ,255  ,127  ,127  ,127  ,127  ,
C    *           127  ,127  ,127  ,127  ,127  ,127  ,127  ,127  ,127  ,
C    *           127  ,127  ,127  ,127  ,501  ,-1   /
C DAS ENTSPRICHT/8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,7BIT ,7BIT ,7BIT ,7BIT ,
C                7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,
C                7BIT ,7BIT ,7BIT,7BIT,9BIT,17BIT/
C DAS ENTSPRICHT/8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,7BIT ,7BIT ,7BIT ,7BIT ,
C                7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,7BIT ,
C                7BIT ,7BIT ,7BIT,7BIT,9BIT,17BIT/
C 8. MASKENSATZ
C
C     DATA MASKE/ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,
C    *           ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,ZFF00,
C    *           ZFF00,ZFF00,ZFF00,ZFF00,ZFE00,ZFFFF/
C     DATA WEITE/255  ,255  ,255  ,255  ,255  ,255  ,255  ,255  ,255  ,
C    *           255  ,255  ,255  ,255  ,255  ,255  ,255  ,255  ,255  ,
C    *           255  ,255  ,255  ,255  ,501  ,-1   /
C DAS ENTSPRICHT/8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,
C                8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,8BIT ,
C                8BIT ,8BIT ,8BIT,8BIT,9BIT,18BIT/
C     ZAHL DER       |       |
C   MASKIERTEN BITS | MASKE | WEITE
C -----------------------------------
C       16  BIT      | Z0000 | 65535
C       15  BIT      | Z8000 | 32767
C       14  BIT      | ZC000 | 16383
C       13  BIT      | ZE000 |  8191
C       12  BIT      | ZF000 |  4095
C       11  BIT      | ZF800 |  2047
C       10  BIT      | ZFC00 |  1023
C        9  BIT      | ZFE00 |   501
C        8  BIT      | ZFF00 |   255
C        7  BIT      | ZFF80 |   127
C        6  BIT      | ZFFC0 |    63
C        5  BIT      | ZFFE0 |    31
C        4  BIT      | ZFFF0 |    15
C        3  BIT      | ZFFF8 |     7
C        2  BIT      | ZFFFC |     3
C        1  BIT      | ZFFFE |     1
C        0  BIT      | ZFFFF |     0
      END
C
C
C
      COMMON/CONST/MIN,AUSGAB,INDSET,IND,TWOPI,MAXVER,VERGEB
      COMMON/CAMM/MASKE(24),WEITE(24)
      COMMON/REGIST/REG(15),SPURDR,NSPUR
      COMMON/TREF/HIT(50,24)
      COMMON/VAR/SPUR(24),PUNKT(24),TREFFE(24),R,P100
      COMMON/MASK/MASK
      COMMON/BINNS/BINN
      COMMON/START/ANFANG,ANZAHL
      LOGICAL    AUSGAB,SPURDR,ANFANG
      INTEGER*2  MASKE,WEITE,MASK
      INTEGER    L,O,P,BINN,INDSET,IND,VERGEB,MAXVER,SPUR,TREFFE,NSPUR
      REAL       TANGEN,PHI,C,RADIUS,RADIEN,TWOPI,MIN,REG,R,P100,PUNKT
      INTEGER    I,J,K,M,N,ABRUCH
      INTEGER*4 NSEC,IUHR
      REAL*8 FNORM
```

```
      DIMENSION RADIEN(24)
      DATA RADIEN/21.83,22.62,23.44,24.26,25.11,25.96,26.83,27.70
     *,28.58,29.48,30.39,31.30,32.21,33.13,34.06,34.98,35.93,36.86
     *,37.81,38.75,39.71,40.66,41.62,42.52/
      CALL TMLOG(2)
      ANZAHL=0
      SIMUL=0
      K=0
      DO 8 J= 1,80,1
      DO 7 I=1,63,2
      K=K+1
      DO 1 L=1,24
      HIT(K,L) =REAL(I)/10+ASIN(RADIEN(L)/(200*REAL(J)))
C     *         + REAL((FNORM(0)*0.001))
      IF (HIT(K,L).GT.TWOPI) HIT(K,L)=HIT(K,L)-TWOPI
1     CONTINUE
      IF (K.LT.10) GOTO 14
      K=0
      DO 3 L=1,24
      TREFFE(L)=10
3     CONTINUE
      DO 4 L=1,24
      KMIST=RN(0)*23+1
      TREFFE(KMIST)=TREFFE(KMIST)+1
      HIT(TREFFE(KMIST),KMIST)=RN(0)*TWOPI
4     CONTINUE
      CALL P1
14    RADIUS=REAL(J)*100.
C     TANGEN=REAL(I)
C     TANGEN=TANGEN/10
      SIMUL=SIMUL+1
7     CONTINUE
8     CONTINUE
C     WRITE (6,11)
C     WRITE (6,12) (WEITE(K),K=1,12)
C     WRITE (6,13) (WEITE(K),K=13,24)
C 11  FORMAT(' SUCHEWEITE:')
C 12  FORMAT(' 1-12 : ',12I5)
C 13  FORMAT(' 13-24: ',12I5)
      CALL TMLOG(0)
      WRITE(6,606) ANZAHL,SIMUL
 606  FORMAT('  GEFUNDENE SPUREN ',F10.3,' SIMULIERTE SPUREN ',F10.3)
      STOP
      END
C
C ******************************************************************
C ******************** SUBROUTINE P1 *******************************
C ******************************************************************
C
      SUBROUTINE P1
C
      COMMON/CONST/MIN,AUSGAB,INDSET,IND,TWOPI,MAXVER,VERGEB
      COMMON/CAMM/MASKE(24),WEITE(24)
      COMMON/REGIST/REG(15),SPURDR,NSPUR
      COMMON/TREF/HIT(50,24)
      COMMON/VAR/SPUR(24),PUNKT(24),TREFFE(24),R,P100
C
      INTEGER   IND,VERGEB,SPUR,K,NSPUR,
     *          INDSET,MAXVER,TREFFE
C
      REAL      TWOPI,MIN,REG,R,R12,R123,P100,P101,PUNKT,HIT
     *          ,UG20,UG21,UG22,UG23
     *          ,OG20,OG21,OG22,OG23,ALPHA,ALPH1
     *          ,PHI,PHI1,PHI3
C
      LOGICAL   AUSGAB,SPURDR
C
```

```
      INTEGER*2 MASKE,WEITE
C
C  ********************************************************************
C  ***************  BEGINN DER SPURENSUCHE  ***********************
C  ***************        LAGE 24           ***********************
C  ********************************************************************
C
      NSPUR    = 0
      SPUR(24) = 0
C
C  ********************************************************************
C  **** HIERHER WIRD NACH ERFOLGREICH BEENDETER SPURENSUCHE ********
C  ****    GESPRUNGEN, UM EINEN NEUEN VERSUCH ZU STARTEN     ********
C  ********************************************************************
C
241   IF(SPUR(24).GE.TREFFE(24)) GOTO193
C
C  ********************************************************************
C  *********  SPRUNG ZUM PROGRAMM ENDE, FALLS ALLE   ****************
C  ********* PUNKTE AUS DER LAGE 24 BEARBEITET SIND:*****************
C  ********************************************************************
C
      SPUR(24)  = SPUR(24)+1
      PUNKT(24) = HIT(SPUR(24),24)
C
      IF(AUSGAB) WRITE(6,242) SPUR(24),PUNKT(24)
242   FORMAT(//,' BEGIN DER SPURSUCHE',I3,
     *      ' MIT PUNKT(24): ',F10.5)
C
C  ********************************************************************
C  ******************    SUCHE IN LAGE 23  ************************
C  ********************************************************************
C
      CALL CAMASS(PUNKT(24),MASKE(23),WEITE(23),UG23,OG23)
      SPUR(23)=0
C
      IF(AUSGAB) WRITE(6,231) UG23,OG23,SPUR(24)
231   FORMAT(' **** SUCHBEREICH IN LAGE23:',11X,2F10.5,I3)
C
C---------- RUECKSPRUNG LAGE 23
C
232   CALL CAM(23,SPUR(23),TREFFE(23),UG23,OG23,PUNKT(23),*241)
      VERGEB=0
      IND=0
C
      IF (AUSGAB) WRITE(6,233) IND,VERGEB,PUNKT(23),PUNKT(24)
233   FORMAT(' HIT GEFUNDEN IN LAGE 23 IND, VERGEB,PUNKT:',2I3,2F10.5)
C
C  ***************************************************************
C  *********  BERECHNUNG DER RICHTUNG DER SPUR (P101), ************
C  *********       DES SPURADIUS UND DES IN            ************
C  *********      LAGE 22 ERWARTETEN PUNKTES           ************
C  ***************************************************************
C
      IF(PUNKT(23).GT.PUNKT(24)) GOTO 234
C
C  VORZEICHEN '-'
C
      REG(15)=  1.
      ALPHA  =  PUNKT(24)- PUNKT(23)
      PHI1   =  TAB1(ALPHA)
      P101   =  PUNKT(23)- PHI1
      R12    =  TAB2(ALPHA)
      PHI3   =  TAB3(ALPHA)
      P      =  P101 + PHI3
      GOTO 235
C
```

```
C  VORZEICHEN '+'
C
234   REG(15)=  0.
      ALPHA  =  PUNKT(23)- PUNKT(24)
      PHI1   =  TAB1(ALPHA)
      P101   =  PUNKT(23)+ PHI1
      R12    =  TAB2(ALPHA)
      PHI3   =  TAB3(ALPHA)
      P      =  P101 - PHI3
C
235   IF(AUSGAB) WRITE(6,236) PHI1,R12,PHI3,P,P101
236   FORMAT(3X,' PHI1=',F10.4,'    R12=',F10.2,'  PHI3=',F10.5,
     *'  P23=',F10.5,'  P101= ',F10.5)
C
C  ****************************************************************
C  ******************  SUCHE IN LAGE 22  ************************
C  ****************************************************************
C
      CALL CAMASS(P,MASKE(22),WEITE(22),UG22,OG22)
      SPUR(22)=0
C
      IF(AUSGAB) WRITE(6,221) UG22,OG22,P101,(SPUR(K),K=23,24)
221   FORMAT(' **** SUCHBERREICH IN LAGE22:',10X,2F10.5,' P101:',F10.5,
     *' SPUR (23,24):',2I3)
C
C ------> RUECKSPRUNGADRESSE LAGE 22
C
222   CALL CAM(22,SPUR(22),TREFFE(22),UG22,OG22,PUNKT(22),*227)
C
      IF (AUSGAB) WRITE(6,223) SPUR(22),TREFFE(22),PUNKT(22)
223   FORMAT (3X,' SPUR(22), TREFFER(22),PUNKT(22) :',13X,2I6,F10.5)
C
C  ****************************************************************
C  ***************    NEUBERECHNUNG DER RICHTUNG     **************
C  ***************  DER SPUR (P100), DES SPURRADIUS  **************
C  ***************    UND VORRAUSBERECHNUNG DES IN   **************
C  ***************     LAGE 21 ERWARTETEN PUNKTES    **************
C  ****************************************************************
C
      IF(PUNKT(22).GT.PUNKT(24)) GOTO 224
C
C  VORZEICHEN '-'
C
      REG(15) =  1.
      ALPHA   =  PUNKT(24)  - PUNKT(22)
      PHI     =  TAB4(ALPHA)
      P100    =  PUNKT(22)  - PHI
      P100    =  P100/2.    + P101/2.
      ALPH1   =  PUNKT(24)  - P100
      R123    =  TAB5(ALPH1)
      P       =  P100 + PHI
      GOTO 225
C
C  VORZEICHEN '+'
C
224   REG(15) = 0.
      ALPHA   =  PUNKT(22)  - PUNKT(24)
      PHI     =  TAB4(ALPHA)
      P100    =  PUNKT(22)  + PHI
      P100    =  P100/2.    + P101/2.
      ALPH1   =  P100       - PUNKT(24)
      R123    =  TAB5(ALPH1)
      P       =  P100 - PHI
C
225   IF(AUSGAB) WRITE(6,226) PHI3,P100,P101
226   FORMAT(3X,' PHI3=',F10.5,'  P100=',F10.5, ' P101=',F10.5)
      GOTO 2211
```

```
C
C  ****************************************************************
C  ************   EXTRAPOLATION UEBER LAGE 22  ********************
C  ****************************************************************
C
227   IF(IND.GT.INDSET)GOTO 232
      VERGEB=0
      P100 = P101
      R123 = R12
      PUNKT(22)= 9.9999
C
2211  PHI3  = TAB6(R123)
      IF (REG(15).EQ.0.) GOTO 228
      P     = P100 + PHI3
      GOTO 229
228   P     = P100 - PHI3
229   CONTINUE
      IF (AUSGAB) WRITE(6,2210) ALPHA,PHI1,R123,PHI3,P
2210  FORMAT (3X,' ALPHA3=',F10.5,' PHI1=',F10.5,
     *' R123=',F10.2,' PHI4=',F10.5,' P4=',F10.5)
C
C  ****************************************************************
C  ******************  SUCHEN IN LAGE 21  *************************
C  ****************************************************************
C
      IF (IND.NE.0) GOTO 219
      CALL CAMASS(P,MASKE(21),WEITE(21),UG21,OG21)
      SPUR(21)=0
211   IF (AUSGAB) WRITE(6,212) 21,UG21,OG21,P100,SPUR(24),SPUR(21)
212   FORMAT(' SUCHBEREICH IN LAGE ',I2,': ',2F10.5,' P100:',F12.5
     *,' SPUR(24,21):',2I3)
219   CALL CAM(21,SPUR(21),TREFFE(21),UG21,OG21,PUNKT(21),*216)
C
C  ****************************************************************
C  ************** NEUBERECHNUNG DES RADIUS ************************
C  ***************    BERECHNUNG VON PUNKT  ************************
C  ***************        IN LAGE 20        ************************
C  ****************************************************************
C
      IF(PUNKT(21).GT.P100) GOTO 213
C
C   VORZEICHEN '-'
C
      REG(15) = 1.
      ALPH1   = P100-PUNKT(21)
      R       = TAB7(ALPH1)
      GOTO 214
C
C   VORZEICHEN '+'
C
213   REG(15) = 0.
      ALPH1   = PUNKT(21) - P100
      R       = TAB7(ALPH1)
C
214   CONTINUE
215   FORMAT(3X,' PUNKT(21)=',F10.5,' ALPH1 =',F10.5,' R=',F10.2
     *,' PHI=',F10.5,' P=',F10.5)
      GOTO 2110
C
C  ****************************************************************
C  ***********  EXTRAPOLIEREN UEBER LAGE 21 ************************
C  ***********     MARKIEREN VON PUNKT(21)  ************************
C  ****************************************************************
C
216   VERGEB   = VERGEB + 1
      R        = R123
      PUNKT(21)= 9.9999
```

```
2110  PHI1     = TAB8(R)
      IF (REG(15).EQ.0.) GOTO 217
      P        = P100 - PHI1
      GOTO 218
217   P        = P100 + PHI1
218   CONTINUE
C
      IF(AUSGAB) WRITE(6,215) PUNKT(21),ALPH1,R,PHI1,P
C
C  ****************************************************************
C  ******************  SUCHEN IN LAGE 20  *************************
C  ****************************************************************
C
      CALL CAMASS(P,MASKE(20),WEITE(20),UG20,OG20)
      SPUR(20)=0
      IF (AUSGAB) WRITE(6,201) 20,UG20,OG20,P100
201   FORMAT(' SUCHBEREICH IN LAGE ',I2,': ',2F10.5,' P100:',F12.5)
      CALL CAM(20,SPUR(20),TREFFE(20),UG20,OG20,PUNKT(20),*202)
      GOTO 191
C
C  ****************************************************************
C  ***********  EXTRAPOLIEREN UEBER LAGE 20 ************************
C  ***********     MARKIEREN VON PUNKT(20)  ************************
C  ****************************************************************
C
202   VERGEB= VERGEB + 1
      PUNKT(20)= 9.9999
C
C  ****************************************************************
C  ************* SPURSUCHE IN DEN RESTLICHEN **********************
C  *************  KAMMERN IM UNTERPROGRAMM   **********************
C  *************        REST                 **********************
C  ****************************************************************
C  ************* ABNORMAL RETURN: SPUR GEFUNDEN *******************
C  ************* NORMALES RETURN: KEINE SPUR    *******************
C  ****************************************************************
C
191   CALL REST (*192)
C
C  ****************************************************************
C  ************ VERGEBLICH SUCHE, SPRUNG  *************************
C  ************ IN LAGE 22: NEUER VERSUCH!*************************
C  ****************************************************************
C
      VERGEB = 0
      IND        = IND + 1
      GOTO 222
C
C  ****************************************************************
C  ********** ERFOLGREICHE SUCHE. SPURDATEN ***********************
C  ********** BEREITS GESPEICHERT. NAECHSTE ***********************
C  **********    TREFFER IN KAMMER 24       ***********************
C  ****************************************************************
C
192   GOTO 241
C
193   RETURN
      END
C
C  ****************************************************************
C  ****************** ENDE SUBROUTINE  P1    **********************
C  ****************************************************************
C
C-----------------------------------------------------------------
C
C  ****************************************************************
C  ************* ANFANG SUBROUTINE REST ***************************
```

```
C  ******************************************************************
C
      SUBROUTINE REST(*)
C
      COMMON/CONST/MIN,AUSGAB,INDSET,IND,TWOPI,MAXVER,VERGEB
      COMMON/CAMM/MASKE(24),WEITE(24)
      COMMON/REGIST/REG(15),SPURDR,NSPUR
      COMMON/TREF/HIT(50,24)
      COMMON/VAR/SPUR(24),PUNKT(24),TREFFE(24),R,P100
      LOGICAL    AUSGAB,SPURDR
      INTEGER*2  MASKE,WEITE
      INTEGER    INDSET,IND
     *           ,VERGEB,MAXVER,SPUR,TREFFE,NSPUR
      REAL       TWOPI,MIN,REG,R,P100,PUNKT,P,UG,OG
999   FORMAT(' SUCHBEREICH LAGE ',I2,': ',2F10.5,' P100:',F12.5)
888   FORMAT(3X,' PHI=',F10.5,' P100=',F10.5,'P=',F10.5,
     *' R=',F10.5,' PUNKT=',F10.5)
C-----------------------------------------------------------------------
C
C  ******************************************************************
C  ***************** SUCHE IN LAGE  19 ******************************
C  ******************************************************************
C
      PHI1 =  TAB9(R)
      IF(REG(15).EQ.0.) GOTO 191
      P    = P100 - PHI1
      GOTO 192
191   P    = P100 + PHI1
192   CONTINUE
C
      CALL CAMASS(P,MASKE(19),WEITE(19),UG,OG)
      SPUR(19) = 0
C
C
      CALL CAM(19,SPUR(19),TREFFE(19),UG,OG,PUNKT(19),*193)
      GOTO 194
C
193   VERGEB    = VERGEB + 1
      PUNKT(19) = 9.9999
C
194   IF (VERGEB.GE.MAXVER)  RETURN
C
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(19)
      IF (AUSGAB) WRITE(6,999) 19,UG,OG,P100
C
C  ******************************************************************
C  ***************** SUCHE IN LAGE 18 *******************************
C  ******************************************************************
C
      PHI1 =  TAB10(R)
      IF(REG(15).EQ.0.) GOTO 181
      P    = P100 - PHI1
      GOTO 182
181   P    = P100 + PHI1
182   CONTINUE
C
      CALL CAMASS(P,MASKE(18),WEITE(18),UG,OG)
      SPUR(18) = 0
C
C
      CALL CAM(18,SPUR(18),TREFFE(18),UG,OG,PUNKT(18),*183)
      GOTO 184
C
183   VERGEB    = VERGEB + 1
      PUNKT(18) = 9.9999
C
C
```

```
184   IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(18)
      IF (AUSGAB) WRITE(6,999) 18,UG,OG,P100
C
C  ********************************************************************
C  ***************** SUCHE IN LAGE 17 *****************************
C  ********************************************************************
C
      PHI1 =  TAB11(R)
      IF(REG(15).EQ.0.) GOTO 171
      P    = P100 - PHI1
      GOTO 172
171   P    = P100 + PHI1
172   CONTINUE
C
C
      CALL CAMASS(P,MASKE(17),WEITE(17),UG,OG)
      SPUR(17) = 0
C
C
      CALL CAM(17,SPUR(17),TREFFE(17),UG,OG,PUNKT(17),*173)
      GOTO 174
C
173   VERGEB    = VERGEB + 1
      PUNKT(17) = 9.9999
C
174   IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(17)
      IF (AUSGAB) WRITE(6,999) 17,UG,OG,P100
C
C  ********************************************************************
C  ***************** SUCHE IN LAGE 16 *****************************
C  ********************************************************************
C
      PHI1 =  TAB12(R)
      IF(REG(15).EQ.0.) GOTO 161
      P    = P100 - PHI1
      GOTO 162
161   P    = P100 + PHI1
162   CONTINUE
C
C
      CALL CAMASS(P,MASKE(16),WEITE(16),UG,OG)
      SPUR(16) = 0
C
C
      CALL CAM(16,SPUR(16),TREFFE(16),UG,OG,PUNKT(16),*163)
      GOTO 164
C
163   VERGEB    = VERGEB + 1
      PUNKT(16) = 9.9999
C
164   IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(16)
      IF (AUSGAB) WRITE(6,999) 16,UG,OG,P100
C
C  ********************************************************************
C  ***************** SUCHE IN LAGE 15 *****************************
C  ********************************************************************
C
      PHI1 =  TAB13(R)
      IF(REG(15).EQ.0.) GOTO 150
      P    = P100 - PHI1
      GOTO 152
150   P    = P100 + PHI1
152   CONTINUE
C
```

```
C
      CALL CAMASS(P,MASKE(15),WEITE(15),UG,OG)
      SPUR(15) = 0
C
C
      CALL CAM(15,SPUR(15),TREFFE(15),UG,OG,PUNKT(15),*153)
      GOTO 154
C
153   VERGEB    = VERGEB + 1
      PUNKT(15) = 9.9999
C
154   IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(15)
      IF (AUSGAB) WRITE(6,999) 15,UG,OG,P100
C
C  ********************************************************************
C  ***************** SUCHE IN LAGE 14  ********************************
C  ********************************************************************
C
      PHI1 =  TAB14(R)
      IF(REG(15).EQ.0.) GOTO 141
      P    = P100 - PHI1
      GOTO 142
141   P    = P100 + PHI1
142   CONTINUE
C
C
      CALL CAMASS(P,MASKE(14),WEITE(14),UG,OG)
      SPUR(14) = 0
C
C
      CALL CAM(14,SPUR(14),TREFFE(14),UG,OG,PUNKT(14),*143)
      GOTO 144
C
143   VERGEB    = VERGEB + 1
      PUNKT(14) = 9.9999
C
144   IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(14)
      IF (AUSGAB) WRITE(6,999) 14,UG,OG,P100
C
C  ********************************************************************
C  ***************** SUCHE IN LAGE 13 *********************************
C  ********************************************************************
C
      PHI1 =  TAB15(R)
      IF(REG(15).EQ.0.) GOTO 131
      P    = P100 - PHI1
      GOTO 132
131   P    = P100 + PHI1
132   CONTINUE
C
C
      CALL CAMASS(P,MASKE(13),WEITE(13),UG,OG)
      SPUR(13) = 0
C
C
      CALL CAM(13,SPUR(13),TREFFE(13),UG,OG,PUNKT(13),*133)
      GOTO 134
C
133   VERGEB    = VERGEB + 1
      PUNKT(13) = 9.9999
C
134   IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(13)
      IF (AUSGAB) WRITE(6,999) 13,UG,OG,P100
C
```

```
C  ******************************************************************
C  ***************** SUCHE IN LAGE 12  ******************************
C  ******************************************************************
C
      PHI1 =  TAB16(R)
      IF(REG(15).EQ.0.) GOTO 121
      P    = P100 - PHI1
      GOTO 122
121   P    = P100 + PHI1
122   CONTINUE
C
C
      CALL CAMASS(P,MASKE(12),WEITE(12),UG,OG)
      SPUR(12) = 0
C
C
      CALL CAM(12,SPUR(12),TREFFE(12),UG,OG,PUNKT(12),*123)
      GOTO 124
C
123   VERGEB    = VERGEB + 1
      PUNKT(12) = 9.9999
C
124   IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(12)
      IF (AUSGAB) WRITE(6,999) 12,UG,OG,P100
C
C  ******************************************************************
C  ***************** SUCHE IN LAGE 11 *******************************
C  ******************************************************************
C
      PHI1 =  TAB17(R)
      IF(REG(15).EQ.0.) GOTO 111
      P    = P100 - PHI1
      GOTO 112
111   P    = P100 + PHI1
112   CONTINUE
C
C
      CALL CAMASS(P,MASKE(11),WEITE(11),UG,OG)
      SPUR(11) = 0
C
C
      CALL CAM(11,SPUR(11),TREFFE(11),UG,OG,PUNKT(11),*113)
      GOTO 114
C
113   VERGEB    = VERGEB + 1
      PUNKT(11) = 9.9999
C
114   IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(11)
      IF (AUSGAB) WRITE(6,999) 11,UG,OG,P100
C
C  ******************************************************************
C  ***************** SUCHE IN LAGE 10  ******************************
C  ******************************************************************
C
      PHI1 =  TAB18(R)
      IF(REG(15).EQ.0.) GOTO 101
      P    = P100 - PHI1
      GOTO 102
101   P    = P100 + PHI1
102   CONTINUE
C
C
      CALL CAMASS(P,MASKE(10),WEITE(10),UG,OG)
      SPUR(10) = 0
C
```

```
C
      CALL CAM(10,SPUR(10),TREFFE(10),UG,OG,PUNKT(10),*103)
      GOTO 104
C
103   VERGEB    = VERGEB + 1
      PUNKT(10) = 9.9999
C
104   IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(10)
      IF (AUSGAB) WRITE(6,999) 10,UG,OG,P100
C
C  ********************************************************************
C  ***************** SUCHE IN LAGE 9   ****************************
C  ********************************************************************
C
      PHI1 =  TAB19(R)
      IF(REG(15).EQ.0.) GOTO 91
      P    = P100 - PHI1
      GOTO 92
91    P    = P100 + PHI1
92    CONTINUE
C
C
      CALL CAMASS(P,MASKE(9),WEITE(9),UG,OG)
      SPUR(9) = 0
C
C
      CALL CAM(9,SPUR(9),TREFFE(9),UG,OG,PUNKT(9),*93)
      GOTO 94
C
93    VERGEB   = VERGEB + 1
      PUNKT(9) = 9.9999
C
94    IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(9)
      IF (AUSGAB) WRITE(6,999) 9,UG,OG,P100
C
C  ********************************************************************
C  ***************** SUCHE IN LAGE  8  ****************************
C  ********************************************************************
C
      PHI1 =  TAB20(R)
      IF(REG(15).EQ.0.) GOTO 81
      P    = P100 - PHI1
      GOTO 82
81    P    = P100 + PHI1
82    CONTINUE
C
C
      CALL CAMASS(P,MASKE(8),WEITE(8),UG,OG)
      SPUR(8) = 0
C
C
      CALL CAM(8,SPUR(8),TREFFE(8),UG,OG,PUNKT(8),*83)
      GOTO 84
C
83    VERGEB   = VERGEB + 1
      PUNKT(8) = 9.9999
C
84    IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(8)
      IF (AUSGAB) WRITE(6,999) 8,UG,OG,P100
C
C  ********************************************************************
C  ***************** SUCHE IN LAGE  7  ****************************
C  ********************************************************************
C
```

```
      PHI1 =  TAB21(R)
      IF(REG(15).EQ.0.) GOTO 71
      P    = P100 - PHI1
      GOTO 72
71    P    = P100 + PHI1
72    CONTINUE
C
C
      CALL CAMASS(P,MASKE(7),WEITE(7),UG,OG)
      SPUR(7) = 0
C
      CALL CAM(7,SPUR(7),TREFFE(7),UG,OG,PUNKT(7),*73)
C
      GOTO 74
C
73    VERGEB    = VERGEB + 1
      PUNKT(7) = 9.9999
C
74    IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(7)
      IF (AUSGAB) WRITE(6,999) 7,UG,OG,P100
C
C  ****************************************************************
C  ***************** SUCHE IN LAGE  6  ****************************
C  ****************************************************************
C
      PHI1 =  TAB22(R)
      IF(REG(15).EQ.0.) GOTO 61
      P    = P100 - PHI1
      GOTO 62
61    P    = P100 + PHI1
62    CONTINUE
C
C
      CALL CAMASS(P,MASKE(6),WEITE(6),UG,OG)
      SPUR(6) = 0
C
C
      CALL CAM(6,SPUR(6),TREFFE(6),UG,OG,PUNKT(6),*63)
      GOTO 64
C
63    VERGEB   = VERGEB + 1
      PUNKT(6) = 9.9999
C
64    IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(6)
      IF (AUSGAB) WRITE(6,999) 6,UG,OG,P100
C
C  ****************************************************************
C  ***************** SUCHE IN LAGE  5  ****************************
C  ****************************************************************
C
      PHI1 =  TAB23(R)
      IF(REG(15).EQ.0.) GOTO 50
      P    = P100 - PHI1
      GOTO 52
50    P    = P100 + PHI1
52    CONTINUE
C
C
      CALL CAMASS(P,MASKE(5),WEITE(5),UG,OG)
      SPUR(5) = 0
C
C
      CALL CAM(5,SPUR(5),TREFFE(5),UG,OG,PUNKT(5),*53)
      GOTO 54
C
```

```
53    VERGEB   = VERGEB + 1
      PUNKT(5) = 9.9999
C
54    IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(5)
      IF (AUSGAB) WRITE(6,999) 5,UG,OG,P100
C
C  ******************************************************************
C  ***************** SUCHE IN LAGE  4  ******************************
C  ******************************************************************
C
      PHI1 =  TAB24(R)
      IF(REG(15).EQ.0.) GOTO 41
      P    = P100 - PHI1
      GOTO 42
41    P    = P100 + PHI1
42    CONTINUE
C
C
      CALL CAMASS(P,MASKE(4),WEITE(4),UG,OG)
      SPUR(4) = 0
C
C
      CALL CAM(4,SPUR(4),TREFFE(4),UG,OG,PUNKT(4),*43 )
      GOTO 44
C
43    VERGEB    = VERGEB + 1
      PUNKT(4) = 9.9999
C
44    IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(4)
      IF (AUSGAB) WRITE(6,999) 4,UG,OG,P100
C
C  ******************************************************************
C  ***************** SUCHE IN LAGE  3  ******************************
C  ******************************************************************
C
      PHI1 =  TAB25(R)
      IF(REG(15).EQ.0.) GOTO 31
      P    = P100 - PHI1
      GOTO 32
31    P    = P100 + PHI1
32    CONTINUE
C
C
      CALL CAMASS(P,MASKE(3),WEITE(3),UG,OG)
      SPUR(3) = 0
C
C
      CALL CAM(3,SPUR(3),TREFFE(3),UG,OG,PUNKT(3),*33 )
      GOTO 34
C
33    VERGEB    = VERGEB + 1
      PUNKT(3) = 9.9999
C
34    IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(3)
      IF (AUSGAB) WRITE(6,999) 3,UG,OG,P100
C
C  ******************************************************************
C  ***************** SUCHE IN LAGE  2  ******************************
C  ******************************************************************
C
      PHI1 =  TAB26(R)
      IF(REG(15).EQ.0.) GOTO 21
      P    = P100 - PHI1
      GOTO 22
```

```
21    P    = P100 + PHI1
22    CONTINUE
C
C
      CALL CAMASS(P,MASKE(2),WEITE(2),UG,OG)
      SPUR(2) = 0
C
C
      CALL CAM(2,SPUR(2),TREFFE(2),UG,OG,PUNKT(2),*23 )
      GOTO 24
C
23    VERGEB    = VERGEB + 1
      PUNKT(2) = 9.9999
C
24    IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(2)
      IF (AUSGAB) WRITE(6,999) 2,UG,OG,P100
C
C  ******************************************************************
C  ***************** SUCHE IN LAGE  1  ******************************
C  ******************************************************************
C
      PHI1 =  TAB27(R)
      IF(REG(15).EQ.0.) GOTO 11
      P    = P100 - PHI1
      GOTO 12
11    P    = P100 + PHI1
12    CONTINUE
C
C
      CALL CAMASS(P,MASKE(1),WEITE(1),UG,OG)
      SPUR(1) = 0
C
C
      CALL CAM(1,SPUR(1),TREFFE(1),UG,OG,PUNKT(1),*13 )
      GOTO 14
C
13    VERGEB    = VERGEB + 1
      PUNKT(1) = 9.9999
C
14    IF (VERGEB.GE.MAXVER)  RETURN
      IF (AUSGAB) WRITE(6,888) PHI1,P100,P,R,PUNKT(1)
      IF (AUSGAB) WRITE(6,999) 1,UG,OG,P100
C
C
C ***************************************************************
C *************   SPEICHERN DER SPUREN IN   *********************
C ************* SUBROUTINE SPURSP (MAX. 20) *********************
C ***************************************************************
C
      IMAX  =  24
      NSPUR = NSPUR + 1
C     IF (NSPUR.GT.20) RETURN 1
      CALL SPURSP(NSPUR)
      RETURN 1
      END
C
C ***************************************************************
C ************** ENDE VON SUBROUTINE REST ***********************
C ***************************************************************
C
C
C ---------------------------------------------------------------
C
C ***************************************************************
C ************** ANFANG SUBROUTINE SPURSP ***********************
C ***************************************************************
```

```
C
      SUBROUTINE SPURSP(ZAEHLE)
C
      COMMON/CONST/MIN,AUSGAB,INDSET,IND,TWOPI,MAXVER,VERGEB
      COMMON/CAMM/MASKE(24),WEITE(24)
      COMMON/REGIST/REG(15),SPURDR,NHIT
      COMMON/TREF/HIT(50,24)
      COMMON/VAR/SPUR(24),PUNKT(24),TREFFE(24),R,P100
      COMMON/BINNS/BINN
      COMMON/START/ANFANG,ANZAHL
      LOGICAL    AUSGAB,SPURDR
      INTEGER*2  MASKE,WEITE
      INTEGER    BINN,INDSET,IND2,
     *           MAXVER,VERGEB,SPUR,TREFFE,ZAEHLE,NHIT,J,I
      REAL       TRK,TWOPI,MIN,REG,R,P100,PUNKT
      DIMENSION TRK(28,20),NHITS(20)
C
C ***************************************************************
C ***************** SPEICHERN DER SPUREN ************************
C ***************************************************************
C
      TRK(1,ZAEHLE) = R
      TRK(2,ZAEHLE) = P100
      NHITS(ZAEHLE) = 0
      DO 1 J = 3,26
      TRK(J,ZAEHLE) = PUNKT(J-2)
      IF (TRK(J,ZAEHLE).LT.TWOPI) NHITS(ZAEHLE) = NHITS(ZAEHLE) + 1
1     CONTINUE
      TRK(27,ZAEHLE) = REG(15)
C
C
C ***************************************************************
C ********* MARKIERE BEREITS BENUTZTE TREFFER *******************
C ***************************************************************
C
2     DO 3 J = 1,23
      HIT(SPUR(J),J)   = HIT(TREFFE(J),J)
      TREFFE(J)        = TREFFE(J) - 1
3     CONTINUE
      VERGEB           =  0
      ANZAHL=ANZAHL + 1
      IF (SPURDR) GOTO 4
      GOTO 6
C
C ***************************************************************
C ************** AUSGABE DER SPURDATEN: *************************
C ***************************************************************
C
4     WRITE(6,5) ZAEHLE,(TRK(I,ZAEHLE),I=1,2),REG(15)
5     FORMAT(/,' *** SPUR ',I2,' GEFUNDENER RADIUS:',F8.1,
     *' RICHTUNG:',F8.4,' VORZEICHEN:',F3.0)
6     RETURN
      END
C
C ***************************************************************
C *************** SUBROUTINE SPURSP ENDE ************************
C ***************************************************************
C
C ---------------------------------------------------------------
C
C ***************************************************************
C ************* SUBROUTINE CAMASS ANFANG ************************
C ***************************************************************
C
C ***************************************************************
C ****************   CAM ASSOTIATION   **************************
C ***************************************************************
```

```
      SUBROUTINE CAMASS(EIN,IMASKE,IWEITE,UG,OG)
C
      COMMON/CONST/MIN,AUSGAB,INDSET,IND,TWOPI,MAXVER,VERGEB
      COMMON/CAMM/MASKE(24),WEITE(24)
      COMMON/REGIST/REG(15),SPURDR,NHIT
      COMMON/TREF/HIT(50,24)
      COMMON/VAR/SPUR(24),PUNKT(24),TREFFE(24),R,P100
      LOGICAL    AUSGAB,SPURDR
      INTEGER    INDSET,IND
     *           ,VERGEB,MAXVER,SPUR,TREFFE,JUG,JOG
      REAL       TWOPI,MIN,REG,R,P100,PUNKT,FAKTOR,UG,OG,EIN
      INTEGER*2  MASKE,WEITE,IMASKE,IPHI,IWEITE,IUG,IOG
C
C        WENN MAN VON EINER ORTSAUFLOESUNG VON .5 MM AUSGEHT,
C        SO HAT MAN MAXIMAL 13363 BINS
C        2 * PI ENTSPRICHT ALSO    (ERWEITERT AUF VOLLE 15 BIT) !!
C        DER FAKTOR BETRAEGT ALSO 163 !!
      DATA FAKTOR/163./
      IF(EIN.LT.1.E-5) GOTO 1
      IPHI = IFIX(EIN*FAKTOR+0.5)
      GOTO2
1     IPHI = 0
2     IUG  = IAND2(IPHI,IMASKE)
      IOG  = IUG+IWEITE
      JOG  = IOG
      JUG  = IUG
      OG   = FLOAT(JOG)/FAKTOR
      UG   = FLOAT(JUG)/FAKTOR
      IF(UG.LT.0)   UG = UG + TWOPI
      IF(OG.LT.0)   OG = OG + TWOPI
      RETURN
      END
C
C ***************************************************************
C ************** ENDE SUBROUTINE CAMASS ************************
C ***************************************************************
C
C ---------------------------------------------------------------
C
C ***************************************************************
C ************** ANFANG SUBROUTINE CAM *************************
C ***************************************************************
C

      SUBROUTINE CAM(KAMMER,SPURR,TREFF,UG,OG,PUNKTT,*)
      COMMON/CONST/MIN,AUSGAB,INDSET,IND,TWOPI,MAXVER,VERGEB
      COMMON/CAMM/MASKE(24),WEITE(24)
      COMMON/REGIST/REG(15),SPURDR,NHIT
      COMMON/TREF/HIT(50,24)
      COMMON/VAR/SPUR(24),PUNKT(24),TREFFE(24),R,P100
      LOGICAL    AUSGAB,SPURDR
      INTEGER*2  MASKE,WEITE
      INTEGER    INDSET,IND,VERGEB,MAXVER,
     *           SPUR,TREFFE,SPURR,TREFF,NHIT
      REAL       TWOPI,MIN,REG,R,P100,PUNKT,UG,OG,PUNKTT
C
C ***************************************************************
C ********************  CAM AUSGABE  ****************************
C ***************************************************************
C
1     SPURR=SPURR+1
      IF(SPURR.GT.TREFF) GOTO 3
      IF(HIT(SPURR,KAMMER).LE.OG.AND.HIT(SPURR,KAMMER).GE.UG) GOTO 2
      GOTO 1
2     PUNKTT = HIT(SPURR,KAMMER)
C
      RETURN
```

```
3     RETURN1
      END
C
C ************************************************************
C ***************** ENDE SUBROUTINE CAM **********************
C ************************************************************
C
C ------------------------------------------------------------------
C
C ********************************************************************
C *************** SIMULATION DER LOOK-UP-TABLES **********************
C ********************************************************************
C
      FUNCTION TAB1(ALP)
      FAKINT=1.
      ALPHAA=ALP/FAKINT
      IF (ALPHAA.LT.1.E-10) ALPHAA=1.E-10
      X1=42.52/(41.62*SIN(ALPHAA))-(COS(ALPHAA)/SIN(ALPHAA))
      X2=1./X1
      PHII=ATAN(X2)
      TAB1=PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB2(ALP)
      FAKINT=1.
      ALPHAA=ALP/FAKINT
      IF (ALPHAA.LT.1.E-10) ALPHAA=1.E-10
      X1=42.52/(41.62*SIN(ALPHAA))-(COS(ALPHAA)/SIN(ALPHAA))
      X2=1./X1
      PHII=ATAN(X2)
      R1=41.62/(2.*SIN(PHII))
      IF(R1.GT.8000.) R1=8000.
      TAB2=R1*FAKINT
      RETURN
      END
C
      FUNCTION TAB3(ALP)
      FAKINT=1.
      ALPHAA=ALP/FAKINT
      IF (ALPHAA.LT.1.E-10) ALPHAA=1.E-10
      X1=42.52/(41.62*SIN(ALPHAA))-(COS(ALPHAA)/SIN(ALPHAA))
      X2=1./X1
      PHII=ATAN(X2)
      R1=41.62/(2.*SIN(PHII))
      Y1=40.66/(2.*R1)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHI4=ATAN(Y2)
      TAB3=PHI4*FAKINT
      RETURN
      END
C
      FUNCTION TAB4(ALP)
      FAKINT=1.
      ALPHAA=ALP/FAKINT
      IF (ALPHAA.LT.1.E-10) ALPHAA=1.E-10
      X1=42.52/(40.66*SIN(ALPHAA))-(COS(ALPHAA)/SIN(ALPHAA))
      X2=1./X1
      PHII=ATAN(X2)
      TAB4=PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB5(PT)
      FAKINT=1.
      PTT=PT/FAKINT
```

```
      IF (PTT.LE.1.E-10) PTT=1.E-10
      R1=42.52/(2.*SIN(PTT))
      IF (R1.GT.8000.) R1=8000.
      TAB5=R1*FAKINT
      RETURN
      END
C
      FUNCTION TAB6(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=39.71/(RR*2.)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB6=PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB7(PT)
      FAKINT=1.
      PTT=PT/FAKINT
      IF (PTT.LT.1.E-10) PTT=1.E-10
      R4=39.71/(2.*SIN(PTT))
      IF (R4.GT.8000.) R4=8000.
      TAB7= R4*FAKINT
      RETURN
      END
C
      FUNCTION TAB8(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=38.75/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB8= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB9(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=37.81/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB9= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB10(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=36.86/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB10= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB11(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=35.93/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
```

```
      PHII=ATAN(Y2)
      TAB11= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB12(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=34.98/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB12= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB13(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=34.06/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB13=PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB14(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=33.13/(RR*2)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB14=PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB15(R)
      FAKINT=1.
      RR =R/FAKINT
      Y1=32.21/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB15= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB16(R)
      FAKINT=1.
      RR =R/FAKINT
      Y1=31.30/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB16= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB17(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=30.39/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
```

```
      TAB17= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB18(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=29.48/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB18= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB19(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=28.58/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB19= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB20(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=27.70/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB20= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB21(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=26.83/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB21= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB22(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=25.96/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB22=PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB23(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=25.11/(RR*2.)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB23=PHII*FAKINT
```

```
      RETURN
      END
C
      FUNCTION TAB24(R)
      FAKINT=1.
      RR =R/FAKINT
      Y1=24.26/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB24= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB25(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=23.44/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB25= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB26(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=22.62/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB26= PHII*FAKINT
      RETURN
      END
C
      FUNCTION TAB27(R)
      FAKINT=1.
      RR=R/FAKINT
      Y1=21.83/(2.*RR)
      IF (Y1.GE.1.) Y1=0.999
      Y2=Y1/SQRT(1.-(Y1**2.))
      PHII=ATAN(Y2)
      TAB27= PHII*FAKINT
      RETURN
      END
//G.LOGPRINT DD SYSOUT=*
/*   ENDE
```

Literatur

[1] H1-Collaboration
June 1985 Letter of Intent for an Experiment at HERA
(Seite 1–2)

[2] H1-Collaboration
June 1985 Letter of Intent for an Experiment at HERA
(Seite 3–53)

[3] H1-Collaboration
March 25, 1985 Technical Proposal for H1-Detector

[4] ZEUS-Collaboration
1985 Technical Proposal for ZEUS-Detector

[5] W.J.Haynes (DESY and Rutherford Appleton Laboratories)
July 29,1985 The Central H1-Data-Acquisition-System

[6] ZEUS Collaboration
June 28,1985 ZEUS – A detector for HERA
Letter of Intent

[7] H.-J.Stuckenberg, P.Schildt
Mai 1978 MONIKA - Prinzip und Hardware
Unterlagen zur Ausschreibung DESY F56

[8] H.-J.Stuckenberg, P.Schildt, N.Wermes
February 1980 MONIKA - An On-Line Track Following
Microprocessor In High Enery Physics Experiments
DESY- Report 80/11

[9] W.Hillen, N.Wermes
January 1983 MONIKA - A Microprocessor For Track Finding
TASSO Note No.245 DESY-Hamburg

[10] P.Schildt
1979 Entwicklung eines mikroprogrammierbaren
Rechners zur on-line Rekonstruktion von Spuren für
den Detektor TASSO am Speicherring PETRA
Diplomarbeit an der Universität Hamburg

[11] H.Kowalski
July 1980 Pattern Recognition In Track Chamber Spectrometers
DESY- Report 80/72 DESY-Hamburg

[12] D.G.Cassel, H.Kowalski
July 1980 Pattern Recognition In Layered Track Chambers Using A Tree Algorithm
DESY- Report 80/107 DESY-Hamburg

[13] H.-J.Stuckenberg
Two Level Triggering In Storage Ring Experiments
CERN- Conference APPL. MICROS TO HEP 1981 CERN 81-07, 17 July 1981
(Seite 460–470)

[14] M.Lorenschat
Februar 1982 Programm zur Spurenkonstruktion für den Mikroprozessor MC68000
Diplomarbeit an der Fachhochschule Wedel

[15] G.Darbo, S.Vitale
April 7, 1981 A Second Stage Trigger For Track Reconstruction Based On Contiguity Ideas
Nuclear Instruments and Methods 190(1981)
(Seite 81–88)

[16] G.Darbo, L.Rossi, G.Musso, L.Stringa, S.Vitale
April 7, 1981 A High Parallesism Structure for Real Time Pattern Recognition Applied To Colliding Beam Experiments
Report of University of Genua
(Seite 189-210)

[17] H.-J.Behrend,V.Schröder
January 1986 A Drift Chamber Trigger For H1 As A First Level Trigger
Report H1-01/86 - 41 DESY Hamburg

[18] I.N.Bronstein K.A.Semendjajew
1981 Taschenbuch der Mathematik
21. Auflage Teubner Verlagsgesellschaft

[19] H.v.Mangold, K.Knopp
1965 Einführung in die höhere Mathematik
12. Auflage S.601-607

[20] P.Waloschek
März 1987, Experimente für HERA
Physikalische Blätter Jahrgang 43, Heft 3
Seite 75–76

[21] P.Waloschek
Februar 1987, Auf der Suche nach den

Urbausteinen unserer Welt
Kosmos Heft 2
Seite 34–40

[22] PR DESY
August 1986 HERA

[23] Heinz Ebert
Oktober 1987, Starke Familienbande- Die Transputerbausteine
C't Magazin für Computer und Technik Heft 10
Seite 86–89,180–188

[24] Product Information by INMOS Limited
March 1986, The Transputer Family

[25] Product Description by INMOS Limited
September 1985, Transputer Architecture

[26] Product Description by INMOS Limited
September 1985, IMS T414 Transputer

[27] Product Description by INMOS Limited
September 1985, IMS T212

[28] Product Description by INMOS Limited
September 1985, IMS B001 Transputer Evaluation Board

[29] Product Description by INMOS Limited
September 1985, IMS C001 Link Adaptor

[30] A.V. Aho, J.E.Hopcraft, J.D.Ullmann *Oktober 1987, The Design and Analysis of Computer Algorithms*
ISBN 0-201-00029-6
Seite 87–92,172–179

[31] Motorola
1984 MC68020 32-Bit Microprocessor User's Manual
ISBN 0-13-541467-9

[32] G.Grindhammer,D.Lüers,R.Mundt,H.Oberlack,P.Ribarics
September 1987 The First Level Liquid Argon Calorimeter Trigger Physics Requirements
H1–TR–300 Internal Report DESY-Hamburg

[33] J.H.Field
Dezember 1985 Triggering the H1 Detector
H1–85/12–38 Internal Report DESY-Hamburg

[34] H.Hofmann
Oktober 1984 HERA Experiments Proceedings Discussion
Background Condition at SPS Colliders
(Seite 467–498)
Desy HERA 85/01 Internal Report DESY-Hamburg

[35] W.Bartel
Oktober 1984 HERA Experiments Proceedings Discussion
(Seite 469 ff)
Desy HERA 85/01 Internal Report DESY-Hamburg

[36] W.Bartel,B.Foster,E.Lohrmann,V.Schroeder,U.Koetz,R.Kose
Juni 1985 Synchrotron Radiation Background at HERA Interaction Regions
Desy HERA 85/15 Internal Report DESY-Hamburg

[37] *April 1979 Proceedings of the Study of an e–p Facility for Europe DESY-Hamburg*
(Seite 341 ff)
Desy 79/48 Internal Report DESY-Hamburg

[38] Report of the Elektron Proton Working Group of ECFA
January 1980 Study on the Proton–Elektron Storage Ring Projekt HERA
Desy HERA 80/01 Internal Report DESY-Hamburg
ECFA 80/42

[39] D.MacGregor, D.Mothersole, B.Moyer
August 1984 The Motorola MC68020
IEEE Micro

[40] Th. Naumann
February 1988 Backward chambers and $b\bar{b}$ production
Desy Private Paper

[41] D.H.Saxon
January 1985 Charged particle tracking at HERA
(Seite 153 ff)
Desy HERA 85/01 Internal Report DESY-Hamburg
Discussion meeting on HERA Experiments

[42] H.-J. Behrend
November 1987 Hardware Triggers Using the Driftchamber of H1 Desy H1-11/87-74 Internal Report DESY-Hamburg

[43] Eisele
1987 Transparencies

Abbildungsverzeichnis

1 Topologie von *ep* Ereignisse 6
2 Aufbau des H1-Detektors 7
3 Skizze der Driftkammern. 10
4 Zellstruktur der zentralen Driftkammer. 11
5 Datenauslese der CJC 13
6 Triggersystem 15
7 Zeitdiagramm der verschiedenen Triggerstufen 17
8 Datenflußbei H1 18
9 R–Z Projektion eines $\gamma g \rightarrow b\bar{b}$ Ereignisses 20
10 R–Φ Projektion eines $\gamma g \rightarrow b\bar{b}$ Ereignisses 21
11 Die Transversalenergie aller geladenen Teilchen mit $\Theta > 5°$ 22
12 Die Transversalenergie des gestreuten Elektrons 23
13 Spurmultiplizität in der CJC und MWPC 24
14 Die Verteilung aller geladenen Teilchen nach der Fragmentation . . . 25
15 Der Transversalimpuls aller geladenen Teilchen nach der Fragmentation 26
16 Verwendete Bezeichnungen 32
17 Verschiedene Bäume 35
18 Funktion des "Kletterns" 36
19 Signal mit Suchkoordinaten 37
20 Darstellung von Spuren im reziproken Raum 40
21 Darstellung von Spuren im realen Raum 41
22 symbolische Darstellung eines Schalterkastens 43
23 R-Φ Projektion eines Ereignisses im Maskenprozessor 44
24 Maske zur Überbrückung inaktiver Kammern 45
25 Maske für schwach gekrümmte Spuren 45
26 Technische Realisierung der "Schalter". 45
27 Aussehen eines begrenzten *footballs* 47
28 Diagramm für Signetic CAM 53
29 Blockflußdiagramm des verwendeten Programmes (Teil 1) 57
30 Blockflußdiagramm des verwendeten Programmes (Teil 2) 58
31 Blockflußdiagramm des verwendeten Programmes (Teil 3) 58
32 Blockflußdiagramm des verwendeten Programmes (Teil 4) 59
33 Kleinste Abstände der Spuren vom Wechselwirkungspunkt 65
34 Auftretende Spurmultiplizität in den Monte–Carlo Daten 66
35 Auftretende Radien der Teilchen in den Monte–Carlo Daten 67
36 Radius der rekonstruierten Spuren (Fall A,B) 69
37 Radius der rekonstruierten Spuren (Fall C) 70
38 Kleinster Abstand zum Wechselwirkungspunkt (Fall A,B,C) 71
39 Zeitlicher Ablauf einer Simulation 73
40 Programmlaufzeiten für Untergrundereignisse (Fall A) 78
41 Programmlaufzeiten für Untergrundereignisse (Fall B) 79

42 Programmlaufzeiten für Untergrundereignisse (Fall C) 80
43 Ein nicht getriggertes *real physics* Ereignis 81
44 Der interne Aufbau des DSP56001 (Bockschaltbid) 93
45 Der interne Aufbau des T414 (Bockschaltbid) 102
46 Flußdiagramm des verwendeten Programmes (Teil 1) 110
47 Flußdiagramm des verwendeten Programmes (Teil 2) 111
48 Flußdiagramm des verwendeten Programmes (Teil 3) 112
49 Flußdiagramm des verwendeten Programmes (Teil 4) 113
50 Flußdiagramm des verwendeten Programmes (Teil 5) 114
51 Flußdiagramm des verwendeten Programmes (Teil 6) 115
52 Flußdiagramm des verwendeten Programmes (Teil 7) 116
53 Flußdiagramm des verwendeten Programmes (Teil 8) 117
54 Flußdiagramm des verwendeten Programmes (Teil 9) 118
55 Flußdiagramm des verwendeten Programmes (Teil 10) 119
56 Flußdiagramm des verwendeten Programmes (Teil 11) 120
57 Flußdiagramm des verwendeten Programmes (Teil 12) 121
58 Flußdiagramm des verwendeten Programmes (Teil 13) 122
59 Flußdiagramm des verwendeten Programmes (Teil 14) 123
60 Flußdiagramm des verwendeten Programmes (Teil 15) 124
61 Flußdiagramm des verwendeten Programmes (Teil 16) 125
62 Flußdiagramm des verwendeten Programmes (Teil 17) 126
63 Flußdiagramm des verwendeten Programmes (Teil 18) 127
64 Flußdiagramm des verwendeten Programmes (Teil 19) 128
65 Flußdiagramm des verwendeten Programmes (Teil 20) 129
66 Flußdiagramm des verwendeten Programmes (Teil 21) 130
67 Flußdiagramm des verwendeten Programmes (Teil 22) 131
68 Flußdiagramm des verwendeten Programmes (Teil 23) 132

Tabellenverzeichnis

1 Parameter von HERA 4
2 Raten physikalisch relevanter Ereignisse 29
3 Raten der Untergrundereignisse 30
4 Eigenschaften der Generatoren 63
5 Gewählte Parameter für die Simulation 68
6 Ergebnisse der Simulation 1.Teil 75
7 Ergebnisse der Simulation 2.Teil 76
8 Kosten der Prozessorkarte 83
9 Kosten des CAM–Speichers 83
10 Kosten der Schieberegister–Karten 84
11 Kosten der Winkelberechnungs–Einheiten 85
12 Gesamtkosten des Systems (verschiedene Konfigurationen) 86

10 Zusammenfassung 195

13 Datendarstellung in den Registern 94

Index

Adreß Modifizier Register 95
Adreß–Berechnungs–Einheit 49
Adreßregister R0–R7 95
Analog–Digital–Wandler 12
Assotiation 52
Asynchroner Bus 50
Auflösung der CJC 59
AUSWERT.COM 72

Baum 35
Beam–gas Ereignisse 27
Blätter 35
Bootstrap 92

CAM 52
CAM- Simulationsroutine 54
CAU 49
CJC–Segmente 10
CJC 5 10
  Draht Lagen 10
  Inaktivität der CJC 32
  Segmente 10
  Superzellen 10
COMAND.COM 72
CUTS 72

Datenbus 92
  G- Datenbus 92
  Programmdatenbus 92
  X- Datenbus 92
  Y- Datenbus 92
Datenregister 95
DAU 92
Depth first search 36
Digitaler Signal Prozessor 49
Direkte Funktionen 104
Diskriminator 12
Distance of closest approach 64
Driftstrecken 10

Eichler Generator 61
Ereignisraten 15
EVENT4A.COM 72

Feldspule 5
Football 46
FRITIOF 61

G- Datenbus 92
GEANT 61
GEANT- Generatoren 61
  Eichler Generator 61
  FRITIOF Generator 61
GEP 72
Grundbaum 35

Hadron Kalorimeter 5
Harvard Struktur 49
Hash- Kodierung 52
HERA 4
HISTOGR 72
Host Interface 49

IEEE 754 98
IMS C0xx 104
Inaktivität der CJC 32
Indirekte Funktionen 104
Instruction Cache 50
Interupts 98

Kalorimeter 5
KERMIT 72
Kreisgleichung 32
Kreisverwandtschaft 40

Link in Baumstrukturen 35
Links der Transputer 51
LOOKUP4a.COM 56
Lookup–Tabellen 33
Lost Elektrons 27
Lost Protons 27
Luminosität 4

Maske 42,52

MC-Daten 61
MERGED.COM 72
Monte- Carlo- Daten 61
Multitask 50
Multiplizität eines Ereignisses 64
MWPC 5
Myonkammer 5

Nibble 104

OCCAM 105
Off momentum elektrons 27
Op–Code 106

Pipeline 98
Prefetch 98
Program Counter 95
Programmzähler 95
Programmdatenbus 92

R–Phi-Ebene 31
RAM 52,92
REDUCT 72
Referenzkammer 31
Register
  Adreß Modifizier Register 95
  Adreßregister (R0–R7) 95
  Datenregister 95
  Register (R0–R7) 95
  Register (M0–M7) 95
  Register (X0,X1,Y0,Y1) 95
  Schleifen- Adreß- Register 95
  Statusregister 95
Reziproke Koordinaten 40
RISC–Struktur 49
ROM 92
RUN.CMD 72

SBG 27
Schalterkasten 42,43
Scheduler 50
Schleifenzähler 95
Schleifen- Adreß- Register 95
Speicherblöcke
  X- Speicher 92
  Y- Speicher 92
Schwerpunktsenergie 4
SIM56000 96
Simulator 96
Spallation Protons 27
Spurbögen 31
Stack 95
Stamm 35
Statusregister 95
Strahlenergie 4
Superzelle 10
Supraleitende Magnetspule 12
Synchroton- Strahlung 27

Tabellen- Kodierung 52
Tailcatcher 5
Topolgie der Ereignisse 4
Transputer 50
Triggerstufen 15

UBG 27
Umlauffrequenz 4
Untergrundereignisse 27
Upstream Beam–Gas Collision 27

Vakuum 4
VERTEX 72

Workspace 101
WVW.BAT 72

X- Datenbus 92
X- Speicher 92

Y- Speicher 92
Y- Datenbus 92

Z–Kammern 5
Zellstruktur 11
Zentrale Driftkammer 5

Dank

Ich danke H.-J. Stuckenberg für seine Bemühungen bei der Betreuung dieser Diplomarbeit. Desweiteren danke ich H.-J. Behrend, H. Spitzer und P. Steffen für die Diskussion der Arbeit. Dank auch an Th. Wolff, der bei einigen praktischen Arbeiten mir hilfreich zur Seite stand. Als letztes noch Dank an meine Eltern, welche mich bei der Erstellung dieser Arbeit auf Ihre Weise unterstützten.