FACHHOCHSCHULE DORTMUND

8. Jahrestagung des Siemens-Anwender-Kreises S A K 1977

- Tagungsbericht -

Fachhochschule Dortmund
in Schlagworten

Gegründet am 1. August 1971
Aufgabe: Praxisbezogene Ausbildung auf wissenschaftlicher oder künstlerischer Grundlage.
Studiendauer: 6 Semester, im Fachbereich Design 8 Semester.

4315 Studenten
190 Lehrende
175 Mitarbeiter

Fachbereiche (FB) mit Studiengängen bzw. -richtungen und Abschlüsse

FB Architektur
Architektur (Hochbau)
Städtebau und Landesplanung
Abschluß: Ingenieur (grad.)

FB Design
Produkt - Design mit Schwerpunkt Objekt - Design
Visuelle Kommunikation mit Schwerpunkt Grafik - Design, Foto-/Film- Design.
Abschluß: Designer (grad.)

FB Elektrische Energietechnik
Elektrische Energietechnik
Abschluß: Ingenieur (grad.)

FB Informatik
Allgemeine Informatik
Ingenieurinformatik
Abschluß: Informatiker (grad.)

FB Maschinenbau
Konstruktionstechnik
Fertigungstechnik
Stahlbau
Werkstofftechnik
Abschluß: Ingenieur (grad.)

FB Nachrichtentechnik
Nachrichtentechnik
Abschluß: Ingenieur (grad.)

FB Sozialarbeit
Sozialarbeit
Abschluß: Sozialarbeiter (grad.)

FB Sozialpädagogik
Sozialpädagogik
Abschluß: Sozialpädagoge (grad.)

FB Wirtschaft
Wirtschaft
Abschluß: Betriebswirt (grad.)

Inhaltsverzeichnis

Grußwort des Rektors der Fachhochschule Dortmund

Es ist mir eine Freude, Sie, die Teilnehmer der diesjährigen SAK-Tagung, in der Fachhochschule Dortmund begrüßen zu dürfen. Sie wissen, daß sich die Fachhochschulen allgemein um eine ständige Erneuerung und nach Möglichkeit auch Intensivierung ihres Praxisbezuges bemühen.
Die Fachhochschule Dortmund hat darüber hinaus unmittelbare Kontakte mit der "Abnehmerseite" aufgenommen und sucht durch ständiges Gespräch sowohl hinsichtlich der Hochschullehrer als auch der technischen Mitarbeiter und der Studenten diesem Anliegen gerecht zu werden.
Ich begrüße es deshalb besonders, daß Sie, die Sie doch in einem ganz besonderen Sinne Vermittler zwischen Theorie und Praxis sind, zu Ihrer Zusammenkunft die Fachhochschule Dortmund gewählt haben. Von Ihren Veranstaltungen werden mit Sicherheit Impulse auf die Arbeit in manchen unserer Fachbereiche ausgehen, zumal Sie die Vorträge in den Lehrbetrieb der Fachhochschule zu integrieren beabsichtigen.
Deshalb wünsche ich sehr herzlich Ihrer Tagung einen vollen Erfolg und Ihnen persönlich einen angenehmen Aufenthalt in Dortmund.

W. Strombach

Prof. Dr. W. Strombach

Vorwort

Die SAK-Tagung 1977 in der Fachhochschule Dortmund führte diesmal ca. 130 Teilnehmer zusammen. Erstmals wurden Besichtigungen in den Tagungsablauf eingebaut. Dortmund war hierzu ein geeigneter Standort, da hier eine relativ große Zahl von Siemens-Prozeßrechnern im Einsatz sind. Mit den 18 Vorträgen aus dem Bereich der Prozeßrechentechnik konnte wieder einmal den Mitgliedern des SAK der derzeitige Stand der Technik bzgl. Hardware und Software vermittelt werden. Die regen Diskussionen nach jedem Vortrag zeigte das Interesse und die Notwendigkeit für diese Jahrestagungen. Die Vorträge waren in die Lehrveranstaltungen für Datenverarbeitung an der Fachhochschule integriert; was auch durch den großen Zuhörerkreis - teilweise 200 - sichtbar wurde. Gelegentlich der Hauptversammlung am Donnerstag wurde mit besonderer Genugtuung zur Kenntnis genommen, daß auch der Minister für Wissenschaft und Forschung durch Entsendung eines Vertreters Interesse an dieser Veranstaltung dokumentiert hat.

An dieser Stelle möchte ich allen Mitarbeitern für ihre Hilfe bei der Vorbereitung und Organisation sowie den Herren der Industrie und der Stadtverwaltung Dortmund, die sich um die erfolgreiche Durchführung der Besichtigungen bemühten, herzlich danken.

Darüber hinaus möchte ich auch der Firma Siemens danken, die durch ihr Engagement immer wieder bewiesen hat, wie sehr sie dem SAK verbunden ist.

Prof. Dipl.-Ing. C o s a c k
Tagungsbeauftragter

Besichtigungen während der Jahrestagung

Besichtigt wurden
- das Breitbandwalzwerk der Hoesch AG
- der Verkehrsrechner der Stadt Dortmund
- der Verkehrsüberwachungsrechner der Stadt Dortmund in der Hauptfeuerwache
- das Fg-Werk der Firma Siemens in Witten

Im Breitbandwalzwerk wird eine 306 zur Steuerung, Überwachung und Regelung eingesetzt. Der Prozeßrechner führt die gesamte Verarbeitung in der Vorstraße und in der Fertigstraße. Zuständig ist der Rechner auch für die Steuerung des Temperaturverlaufes in der Kühlstrecke. Weiterhin hat er die Aufgabe der Materialverfolgung und gibt die Bearbeitungsergebnisse an die übergeordneten Rechnersysteme.

Details sind veröffentlicht in der Siemenszeitschrift 47 (73) Beiheft Antriebstechnik und Prozeßautomatisierung in Hütten- und Walzwerken.

Im Rahmen der Besichtigung wurde der im Stadthaus installierte Zentrale-Bedienungs-Rechner (ZBR) Typ 16030 sowie der dort untergebrachte Satellitenrechner für den Bereich südliche Stadtmitte Typ 16004 vom Maschinenamt der Stadt Dortmund vorgestellt. Zur Zeit steuert die 16004 112 Signalanlagen mit 1.750 Signalgruppen . Die Ansteuerung der Anlage erfolgt über ein TST 20-System.

Für Einzelheiten wird auf den abgedruckten Vortrag von Herrn Camen in diesem Heft verwiesen.

Die Prozeßrechner im Fg-Werk-Witten der Siemens AG sind im Bereich Fertigung und Prüfung vom ESK-Cross-Point-Anlagen eingesetzt. Sie haben folgende Aufgaben:
Zeitgerechte und simultane Versorgung der NC-Automaten mit entsprechenden Steuerprogrammen,
Zentrale Verwaltung der Steuerungsprogramme,
Erfassen von automaten-, produkt- und personalbezogenen Daten,
Erfassen von Daten für die Fortschreibung der Betriebsaufträge.
Für diese Probleme ist ein Prozeßrechnersystem mit einer zentralen PR 330 eingesetzt, an die über mehrere PR 310 die Automaten sternförmig angekoppelt sind.

Einzelheiten sind veröffentlicht in telefon report 13 (77), Heft 3.

Demonstrationen an der Prozeßrechner-Anlage 330 der Fachhochschule Dortmund

Am 20.4. - Automatische Skalierung des Analogrechners EAI mit der 330.
Über dieses Programm wurde in der SAK-Mitteilung ...4/76.... berichtet.

Am 21.4. - Datenerfassung am Motorenprüfstand für Verbrennungsmaschinen.
Hierüber wurde während der Jahrestagung berichtet. Der Vortrag ist in diesem Tagungsbericht veröffentlicht.

- Regelung einer elektrischen Antriebsmaschine.

Am 22.4. - Optimierung der Reglerparameter an einem simulierten Regelkreis.
Dieses Problem wurde während der Jahrestagung in Ulm vorgestellt.
Die Weiterentwicklung des Programms ist in der SAK-Mitteilung ..4/76..... veröffentlicht.

FACHHOCHSCHULE DORTMUND

Presseinformation

Dortmund, 15. April 1977

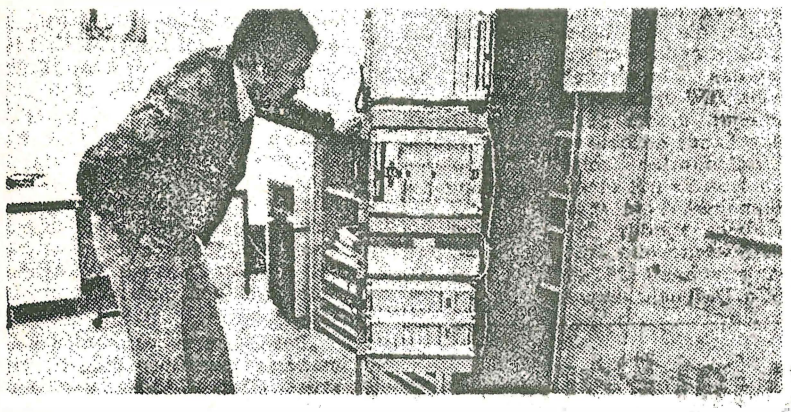
Der Siemens-Prozeßrechner-Anwenderkreis (SAK) führt seine 8. Jahrestagung in der Zeit vom 20. bis 22. April 1977 in der Fachhochschule Dortmund durch.

Dem SAK gehören Unternehmen und Einrichtungen der öffentlichen Hand an, die Prozeßrechner zur Steuerung einsetzen.
Aufgabe des SAK ist der Austausch von Erfahrungen, z. B. Austausch der Programme unter den Anwendern. Die Jahrestagung 1977 wird von Prof. Dipl.-Ing. Cosack vom Fachbereich Nachrichtentechnik der Fachhochschule ausgerichtet.

Aus diesem Anlaß findet am

Mittwoch, dem 20.4.1977, 13.30 Uhr

in Raum 4.1.01, Sonnenstraße 96, eine Pressekonferenz statt, zu der ich Sie herzlich einlade.



130 Computerexperten informieren sich

Seine 8. Jahrestagung führt der Siemens-Prozeßrechner-Anwenderkreis (SKA) bis zum 22. April in der Fachhochschule Dortmund durch. 130 Computerfachleute hören sich 18 Referate an. Außerdem wird die Gruppe unter ihrem Vorsitzenden Prof. Pieper (Ulm) den Verkehrsrechner der Stadt, das Breitbandwalzwerk der Hoesch AG und ein Werk für Telefonanlagen besichtigen. Die Tagung dient auch dem Austausch neuer Programme.

Foto: Thielbeer

A95

RN v. 16.4.

Computer-Experten tagen in der FH

130 Fachleute für Prozeßrechner erwartet die Fachhochschule in der nächsten Woche. Vom 20. bis 22. April führt der Siemens-Prozeßrechner-Anwenderkreis (SAK) seine Jahrestagung durch.

Mitglieder des SAK sind vor allem Hochschulen und Forschungseinrichtungen, aber auch Industriefirmen. Die Mitglieder sind verpflichtet, ihre Erfahrungen auszutauschen. So werden neue Programme an andere Anwender kostenlos abgegeben.

Prozeßrechner sind Computer, die Maschinen und Systeme steuern. So ist der bekannte Verkehrsrechner der Stadt Dortmund zur Steuerung der Ampelanlagen ein Gerät dieses Typs.

WAZ 16.2.77

nformations- und Pressestelle der Fachhochschule Dortmund
4600 Dortmund 1, Sonnenstraße 96, Tel. 12 30 31

Verantwortlich: Hubert Rademacher

Vortragskalender

Herr Fischer — Fachhochschule Dortmund

Konfiguration der Prozeßrechneranlage 330 an der Fachhochschule Dortmund

Herr Abend — Hahn-Meitner-Institut, Berlin

Ein prozeßrechnergesteuertes Strahlenschutz-Überwachungssystem für das Hahn-Meinert-Institut

Herr Strelow — Fachhochschule Dortmund

Automatisierung eines Motorprüfstandes für Verbrennungsmotoren mit einem Siemens Prozeßrechner System 330

Herr Skarek — CERN Genf

Datenerfassung und Datenauswertung mittels Siemens 301 am CERN Synchrozyklotron

Herr Hochmuth — Dornier System GmbH, Friedrichshafen

Der Prozeßrechner Siemens 310 als telefonisches Fahrplanauskunftsterminal mit individuellem Benutzerdialog

Herr Bamberger — Siemens, Karlsruhe

MASK-Software für die 310

Herr Weise — PTB Braunschweig

Das neue Prozeßrechner-Verbundsystem für Meßaufgaben der Physikalisch-Technischen-Bundesanstalt Braunschweig

(Vortrag wird nicht veröffentlicht)

Herr Witte — Siemens, Erlangen

Neue Kopplungsmöglichkeiten mit dem Siemens System 300

(nur Kurzfassung veröffentlicht)

Herr Degelow — Siemens, Karlsruhe

Stand der PEARL - Entwicklung

Herr Bachner — Kernforschungsanlage Jülich

Benutzerfreundliche Bedienung für die Plottersoftware

Herr Schuster — PTB Braunschweig

Ein Prozeßsignalformer zur Vereinfachung der Prozeßeinheit 3600 für Prozeßrechner System 300-16 Bit

Herr John — TH Aachen

Kurvensichtgerät für eine DVA 301

Herr Kirsten — TH Aachen

Cross-Assembler M 6800 für eine DVA 301 bis 305

Herr Janssen — THSchule Aachen

Anschluß eines Plattenspeichers CD - 564 C an eine DVA 301

Herr Huppertz u.a. Kernforschungsanlage Jülich

Ein 8k - Halbleiterspeichermodul für Siemens System 300-16 Bit

Herr Röttgermann Dr. Hell Kiel

Rechner 300 im Einsatz in der Satztechnik

Herr Camen Siemens ZN Dortmund

Prozeßrechner steuert den Verkehr

Herr Krumm u.a. Universität Karlsruhe

Implementierung eines Prozeß-Interpreters (PRINT) auf dem Rechner S 330.

Konfiguration der Prozeßrechneranlage 330 an der Fachhochschule Dortmund

Dipl.-Ing. F i s c h e r, wissenschaftlicher Mitarbeiter an der Fachhochschule Dortmund

Sehr geehrte Damen und Herren,

ich möchte Ihnen an dieser Stelle einen kurzen Überblick über die Konfiguration unserer Prozessrechneranlage Siemens 330 geben.
Im Verlauf dieser Tagung sollen einige Versuche vorgeführt werden, bei denen der Prozessrechner die Meßwerterfassung und zum Teil auch die Regelung übernimmt.
Deshalb will ich hier die Hardware dieser Versuche, soweit sie den Prozessrechner mit der CAMAC-Prozessperipherie betrifft, kurz vorstellen.

DIA I
=====

Die Zentraleinheit ist bei uns zur Zeit bis auf 40 K-Worte (Kernspeicher) ausgebaut.
Diese Tatsache bedingt relativ lange Übersetzungszeiten, insbesondere bei dem Makroübersetzer.
Der Laufbereich von I6 K erlaubt nicht die Benutzung der schnelleren Version, dasgleiche gilt auch für andere Übersetzer, zum Beispiel den Fortran-Compiler.
Die Beschaffung von 24 K-Kernspeicher zur Ergänzung bis zum Vollausbau auf 64 K-Kernspeicher ist aber eingeleitet, so daß dieser Engpaß hoffentlich bald nicht mehr besteht.

DIA 2
=====

Mit dem Wartungsfeld ist die direkteste Einwirkungsmöglichkeit auf den Rechner gegeben. Hardware-Funktionen und Programme können hiermit getestet werden.

DIA 3
=====

Die Standardperipherie des Prozessrechners sieht folgendermaßen aus:
Der EA-Blattschreiber 39II ist als Bedienungseinheit an dem Prozessrechner 330 angeschlossen. Über ihn erfolgt der Dialogverkehr mit der Zentraleinheit.

DIA 4
=====

Der Drucker 39I9 wird zur Protokollierung von Prozessdaten aus dem zu überwachenden Prozess sowie zum Ausdruck von Speicherinhalten und zum Protokollieren bei dem Übersetzen von Programmen verwendet.
Er druckt 200 Zeilen in der Minute.

-II-

DIA 5
=====

Die Lochstreifen Ein- und Ausgabe, hier der Stanzer, dient zum Erstellen von 5- bzw. 8- spurigen Lochstreifen mit maximal 30 Zeichen/sec.
Der Leser kann diese Lochstreifen mit maximal I20 Zeichen/sec lesen.

DIA 6
=====

Der Lochkartenleser 395I ist das an diesem Prozessrechner am häufigsten verwendete Eingabegerät für Programme. Es werden 500 Karten/min. gelesen.

DIA 7
=====

Der Plattenspei cher 394I mit wahlfreiem Zugriff dient zur Erweiterung des Zentralspeichers, da er mit einer Mittleren Datenrate von I23 000 Wörtern/sec. das schnellste Speichergerät ist und über eine Speicherkapazität von I0 Megabytes verfügt.

Licht an.

Dies war ein kurzer Überblick über die Standardperipherie unseres Prozessrechners. Die Prozessperipherie ist hier in Dortmund ausschließlich CAMAC-Peripherie. Sie ist modular aufgebaut und ist dadurch flexibel und anpassungsfähig genug, um sie leicht an sich öfter ändernde Versuchsaufbauten anpassen zu können.
Durch die Verwendung von Sender- und Empfänger-Bausteinen kann hier die Länge des Branch-Highway, das ist der vertikale Datenweg, die Verbindung vom Rechner zu den vier angeschlossenen Crates, bis zu einem Kilometer betragen.
Der Systemcontroller ist wie hier gezeichnet im Rechnerschrank eingebaut. Er regelt den Datenverkehr zwischen dem Prozessrechner 330 und dem Branch-Highway. Über DMA-Bausteine im Zentralspeicher ist er an die Anschlußstelle 4 in der Zentraleinheit angeschlossen.
Ein CAMAC-Crate ist Träger für die steckbaren CAMAC-Baugruppen.
Jedes Crate hat 25 Einbauplätze.
Der Einbauplatz für den Crate-Controller ist festgelegt.
Er belegt die drei rechten Einbauplätze in einem Crate.
Mit dem Crate-Controller wird der Befehls- und Datenverkehr zwischen Branch-Highway und dem Dataway koordiniert. Der Dataway verbindet die einzelnen Baugruppen eines Crates miteinander.
Er hat Busstruktur.
Die am häufigsten vorkommenden Baugruppen sind erstens, die hier eingezeichneten und schon erwähnten Crate-Controller und die Sender und Empfänger für den Branch-Highway.
Hinzu kommen: Dataway Display, zur Anzeige aller Signale auf dem CAMAC-Dataway. Dabei werden Daten und Befehl, mit dem eine Baugruppe angesprochen wird, abgespeichert.

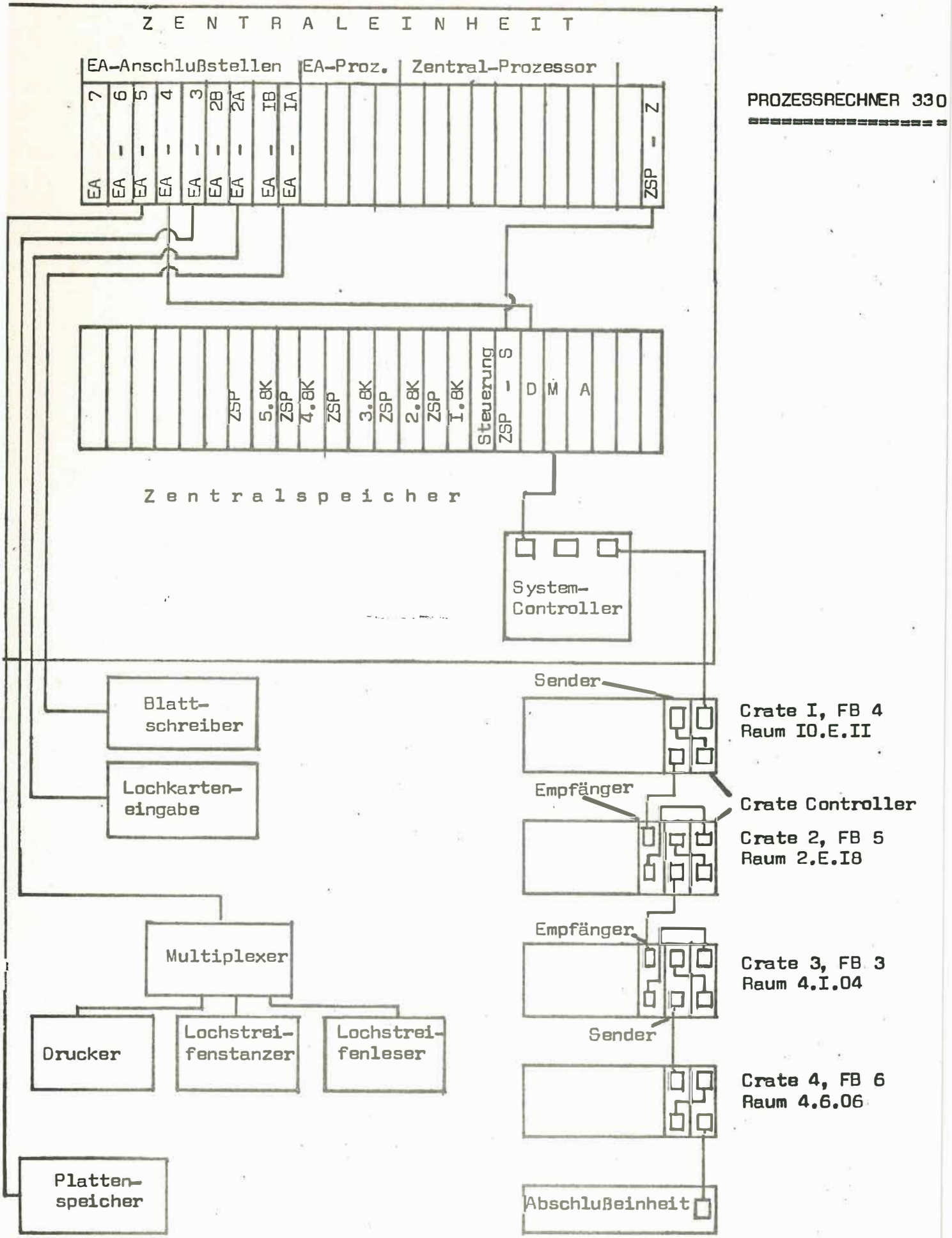
-III-

-III-

Der Multiplexer schaltet durch Befehle analoge Meßgrößen von 32 Meßstellen zweipolig oder von I6 Meßstellen vierpolig durch.
Das Voltmeter wird zum Messen von Gleich- oder Wechselspannungen sowie von Gleichstrom verwendet.
Es entspricht einem Analog-Digitalumsetzer mit Vorverstärker.
Mit einem Analogbus oberhalb des Dataway wird die Verbindung von Voltmeter, Current Source und Multiplexer realisiert.
Current Source wird als Konstantstromquelle bei der Messung von Widerständen und Temperaturen verwendet.
Weitere Baugruppen sind DA/Converter, Clock/Timer, Dynamische und Statische Digitaleingabe sowie Digitalausgabe.
Jetzt werde ich Ihnen kurz die Aufstellungsorte der vier CAMAC-Crates angeben:
Das Crate I steht zusammen mit dem Prozessrechner im Prozessrechnerlabor in Haus IO. Hier soll die Optimierung der Regelparameter an einem simulierten Regelkreis demonstriert werden. Weiterhin wird die automatische Skalierung des Analogrechners mit dem Prozessrechner gezeigt.
Das Crate I wird hauptsächlich durch den Fachbereich 4, Informatik, zum Teil auch durch den Fachbereich 6, Nachrichtentechnik, genutzt.
Das Crate 2 steht bei dem Motorprüfstand im Fachbereich 5, Maschinenbau, hier soll die Erfassung von Messwerten eines Verbrennungsmotors gezeigt werden.
Das Crate 3 steht im Labor für elektrische Maschinen des Fachbereichs 3, elektrische Energietechnik, hier wird eine digitale Lageregelung durchgeführt.
Das Crate 4 steht im Fachbereich 6, Nachrichtentechnik.

Abschließend möchte ich noch etwas über vorgesehene Erweiterungen und Änderungen unserer Konfiguration des Prozessrechners Siemens 330 sagen:
Neben dem schon erwähnten Kernspeicherausbau auf 64 K ist die Beschaffung eines Calcomp Trommelplotters vom Typ 836 vorgesehen.
Um den etwas umständlichen Betrieb der CAMAC-Crates, die nicht im gleichen Raum wie der Rechner stehen, zu erleichtern und sie von der Bedienung her dem bis jetzt hauptsächlich verwendeten Crate I gleichzustellen, ist für I979 die Beschaffung eines Teilnehmerbetriebssystems geplant. Über vier Zeichenbildschirmeinheiten mit Hardcopyeinrichtung, die bei jedem Crate aufgestellt werden sollen, wäre von jedem Crate die gleiche Zugriffsmöglichkeit auf den Rechner gegeben. Ein weiterer großer Vorteil des Teilnehmerbetriebssystems mit vier Terminals ist die Möglichkeit den Rechner für vier Personen gleichzeitig zugänglich zu machen, und ihnen das Erstellen, Übersetzen und Testen von Programmen zu ermöglichen. Die zur Verfügung stehende Arbeitszeit mit dem Rechner wird durch das Teilnehmerbetriebssystem vervielfacht werden.

Ich danke Ihnen für ihre Aufmerksamkeit.


Z E N T R A L E I N H E I T
EA-Anschlußstellen
EA-Proz.
Zentral-Prozessor
7 EA
6 - EA
5 - EA
4 - EA
3 - EA
2B - EA
2A - EA
IB - EA
IA - EA
ZSP - Z
PROZESSRECHNER 330
ZSP
5.8K ZSP
4.8K ZSP
3.8K ZSP
2.8K ZSP
1.8K
Steuerung ZSP - S D M A
Zentralspeicher
System-Controller
Blatt-schreiber
Lochkarten-eingabe
Multiplexer
Drucker
Lochstrei-fenstanzer
Lochstrei-fenleser
Platten-speicher
Sender
Crate I, FB 4
Raum IO.E.II
Empfänger
Crate Controller
Crate 2, FB 5
Raum 2.E.I8
Empfänger
Crate 3, FB 3
Raum 4.I.04
Sender
Crate 4, FB 6
Raum 4.6.06
Abschlußeinheit

Steuerung eines Prüfstandes für Verbrennungsmotoren mit einem Siemens-Prozeßrechner System 300-16 Bit

Klaus - D. Strelow

Versuche an Verbrennungsmotoren erfordern die Aufnahme und Verarbeitung einer Vielzahl von Meßwerten. Bisher wurden diese Meßwerte beobachtet, manuell registriert und verarbeitet. Für gründlichere Untersuchungen ist dieses Verfahren völlig ungeeignet und daher eine rechnergesteuerte Meßwerterfassung und -verarbeitung erforderlich.
Schon seit längerer Zeit findet man teilautomatisierte Prüfstände. Sie sind über folgende weithin verbreitete Steuerungsarten zu betreiben:

1) Zeitplansteuerung
   mit einstellbaren Zeitrelais für Prüfprogramme
   mit begrenzter Stufenzahl

2) Lochstreifensteuerung
   recht hohe reproduzierbare Einstellgenauigkeit

3) Magnetbandsteuerung
   (seltener) Anwendung bisher zur dynamischen oder kurzzeitigen Aufzeichnung von Prüfprogrammen (Beispiel: Anfahren, Straßenfahrten - Straßensimulation)

Wirtschaftliche Vorteile prozeßrechnergesteuerter Prüfstände

1. Weitgehende Entlastung des Prüfpersonals (in unserem Fall der Studenten). Der Prozeßrechner übernimmt die Steuerung und Überwachung des Prüfstandes.
2. Vereinfachung der Dateneingabe und Bedienung durch Klartextabfrage über Ein- / Ausgabe-Blattschreiber.
3. Verringerung der Fehlermöglichkeiten und Ausschalten von Fehlerquellen. Der Rechner erkennt formale Fehler bei der Eingabe und wartet auf Berichtigung.
4. Kürzere Prüfzeiten. (Dadurch nicht zuletzt Kraftstoffersparnis)
5. Verbesserter Schutz für Prüfstand und Prüfling. Der Rechner schaltet bei Überschreiten von zulässigen Toleranzen noch vor auftretenden Störungen die gesamte Anlage ab.

*)

Grundsätzlich soll unterschieden werden zwischen den mit kommerziellen Rechnern betriebenen Prüfständen und den prozeßrechnergesteuerten Prüfständen. Dies kommt einer Unterscheidung in OFF-LINE und ON-LINE-Betrieb gleich. Siehe Bild 1

Beim OFF-LINE-Betrieb müssen hier die Sollwerte von Hand, oder über Zeitplan- bzw. Lochstreifensteuerung vorgegeben und dann die ermittelten analogen Meßwerte digital in den Rechner eingespeichert werden (ebenfalls von Hand). Der Rechner protokolliert Ergebnisse und dient als Datenspeicher. (Zur Not kann man hier einen von den neueren technisch-wissenschaftlichen Taschenrechnern verwenden).
Eine wirlich sinnvolle Automatisierung von Prüfständen erreicht man jedoch nur durch die im ON-LINE-Betrieb arbeitenden Prozeßrechner. Die Vorteile sind z. B. Vorgabe von Sollwerten und Schaltbefehlen oder auch die Überwachung bei unbeaufsichtigtem Betrieb.

Unsere Zielvorstellungen

Da die prozeßautomatisierte Steuerung des an der FH Dortmund installierten Motorprüfstandes erst zu Beginn dieses Semesters in Angriff genommen wurde, kann hier noch keine vollständige Programmbeschreibung geliefert werden. Ich will mich also darauf beschränken unsere Ziele zu formulieren und Ihnen anschließend den Iststand unserer Arbeit mitzuteilen.
Der Rechner soll folgende Funktionen übernehmen:

- Erfassung der Meßwerte mit Abspeicherung und Registrierung
- Vorgabe der Sollwerte

- Steuerung des Prüfstandes (Beispiel: Umschaltung auf Handbetrieb bei Rechnerausfall)
- Überwachung der Absolut- und Grenzwerte, sowie der Betriebsbedingungen (Reglerstellungen, Schalterstellungen usw.)
- Berechnung abgeleiteter Größen aus den registrierten Meßwerten. (Beispiel: Motorleistung aus Drehzahl und dem über die Bremskraft ermittelten Drehmoment; spezifischer Verbrauch während einer bestimmten Anzahl von Umdrehungen bei gleichzeitiger Erfassung des Drehmomentes usw.)
- Darstellung und Protokollierung der Endergebnisse

Hierbei werden die technischen Vorteile des Siemens-Prozeßrechners 330 voll ausgenutzt.

- Die hohe Abfragerate von max. 50 Meßwerten pro Millisekunden und daraus folgend
- die zuverlässige und schnelle Überwachung der Ist- und Sollwerte
- die annähernd gleichzeitige Erfassung von Meßwerten, die dadurch einen optimalen Vergleich ermöglicht
- Auswertung der Meßergebnisse

*)

Diese Ziele erfordern natürlich eine Menge an Arbeit auf Hardware- sowie Software-Seite.
Die Hauptpunkte unserer Arbeit konzentrieren sich auf die zwei hervorgehobenen Blöcke der Übersicht nach Bild 2.
Die Signalaufbereitung und CAMAC-gerechte Aufbereitung der Meßwerte bereitet auf der Hardware-Seite noch Kopfzerbrechen. Es werden zur Zeit im Rahmen von Projektarbeiten die notwendigen speziellen Trennverstärker und Koppeleinheiten erstellt. Hierbei handelt es sich um auf die einzelnen Meßprobleme zugeschnittene Interfacebauteile zwischen den zur Verfügung stehenden Meßvorrichtungen und der vorhandenen CAMAC-Peripherie.
Ebenfalls muß für den umgekehrten Weg die Aufbereitung der vom Rechner gegebenen Steuersignale und deren prüfstandgerechte Weitergabe realisiert werden.
Auf der Software-Seite soll ein Dienstprogramm erstellt werden, das

- Eingriffe in den Ablauf des Versuches von Hand zuläßt
- je nach Aufgabenstellung den Programmablauf für diese spezielle Aufgabenstellung durchläuft
- die o.e. Überwachung auf Grenzwerte durchführt
- die Berechnung verschiedener Motorkenndaten enthält und diese über Drucker und im Falle, daß es sich um Kurven oder Kennlinienscharen handelt über einen Plotter ausgibt.

*)

Istzustand auf der Hardware-Seite:
Die hervorgehobenen Blöcke in Bild 3 sind bereits heute realisiert. Wie Sie erkennen können, handelt es sich zunächst hauptsächlich um Meßwerteingabegeräte. Tatsächlich werden zur Zeit auch lediglich Daten in den Rechner eingelesen und aus diesen die Motorleistung und der spezifische Verbrauch ermittelt. Wenn man bedenkt, daß diese Aufgabe erst seit Anfang März d.J. in Angriff genommen worden ist, so können wir mit den bisherigen Ergebnissen zufrieden sein.

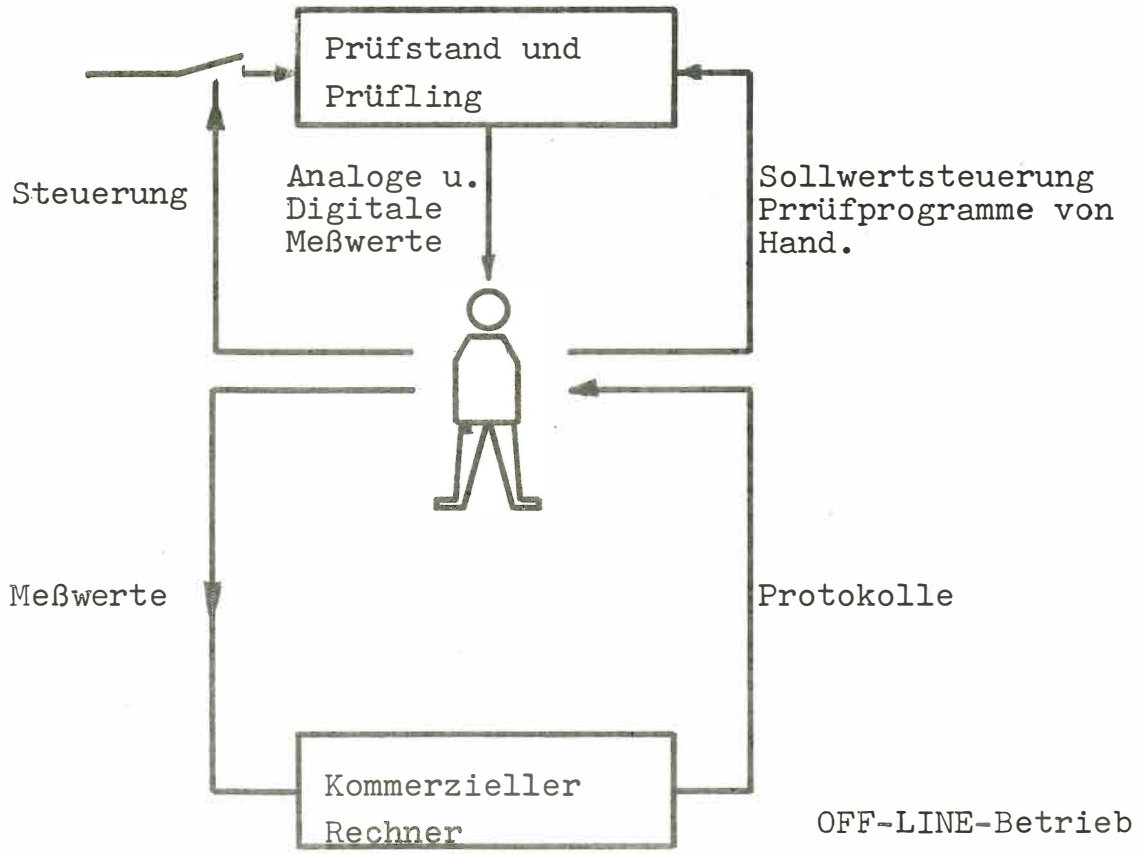
Entsprechend den gerätetechnisch bedingten Möglichkeiten haben wir auf der Software-Seite zunächst ein Programm, das Meßstellen abfragt, Meßwerte einliest und verknüpft. Siehe Bild 4

Bei diesem Programm werden Anzahl und Zeit der Abfrageintervalle und die voraussichliche Versuchsdauer über den Ein- / Ausgabe-Blattschreiber in eine Parametertafel eingeschrieben. Nach Anlegen des Startsignales erfolgt das Einlesen der Realzeit. Die zweite Abfrage bezieht sich auf diese Realzeit. Es wird abgefragt, ob man noch im Einleseintervall für die Abgastemperatur ist, oder ob man bereits im nächsten ist. (Mit NI * I größer als die Realzeit ist man bereits im nächsten Einleseintervall und somit geht das Programm in den rechten Programmzweig).

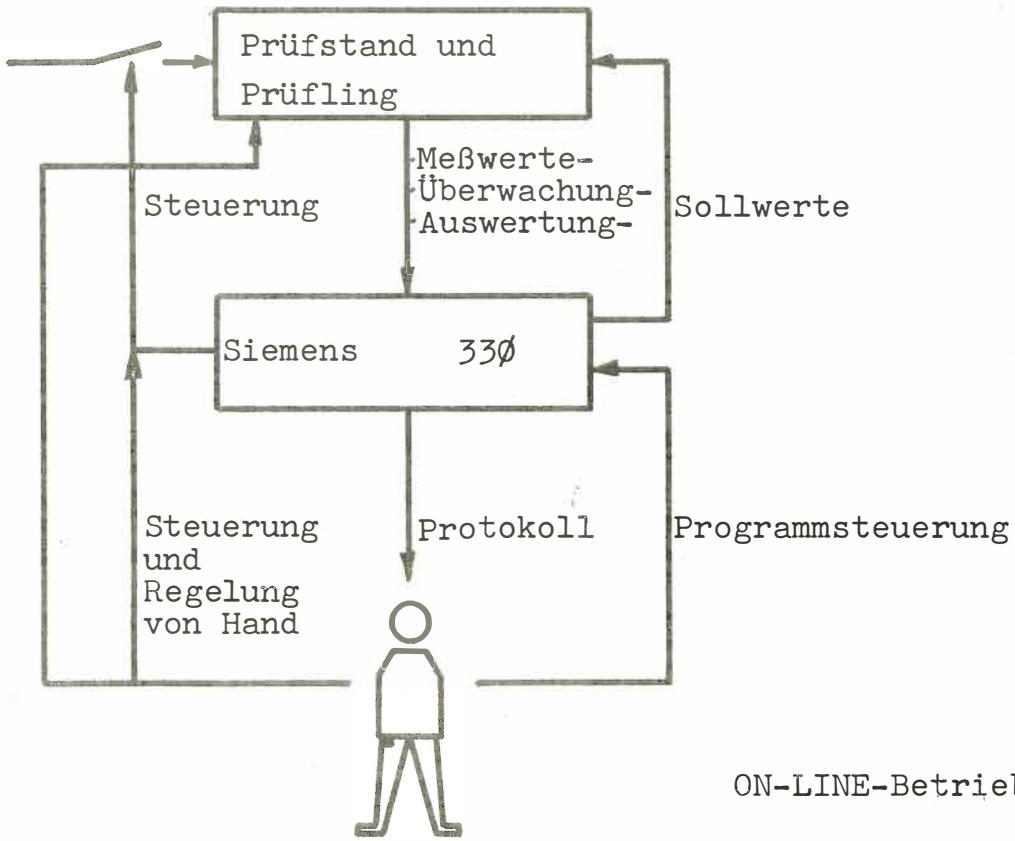
Handelt es sich noch um den 1. Durchlauf, so wird als nächstes die Abfrage nach der halben Gesamtversuchsdauer erfolgen, da Drosselklappenstellung und Bremskraft nur einmal, nach dieser halben Versuchsdauer eingelesen werden. Ist dieser Zweig einmal durchlaufen, so findet keine Meßwerteinlesung für diese beiden Versuchsgrößen mehr statt. Aus dieser Schleife wird das Programm bei Anliegen des Endesignals in die Ausgaberoutine gebracht. Es werden Durchflußzeit, Kraftstoffvolumen und die Anzahl der gemachten Motorumdrehungen eingelesen. Drehmoment, Leistung und spezifischer Verbrauch berechnet und alle Größen textlich aufbereitet und über den Schnelldrucker ausgegeben.
Man sieht, daß bis zur Verwirklichung unserer Vorstellungen noch viel Arbeit vor uns liegt.

Unterschied zwischen rechnerunterstützten und prozeßrechnergesteuerten Motorprüfständen.

-Schematische Darstellung-



OFF-LINE-Betrieb



ON-LINE-Betrieb

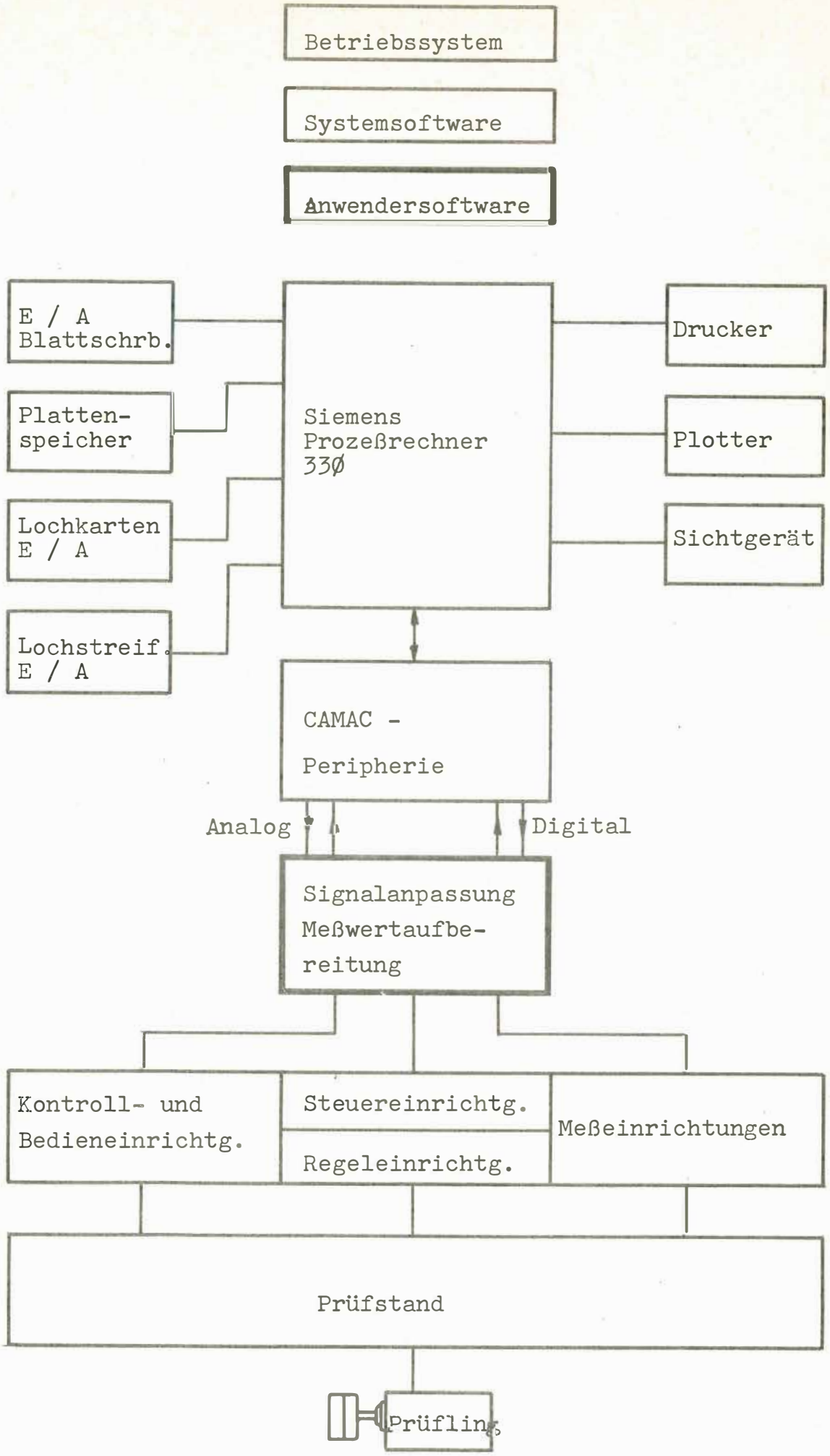
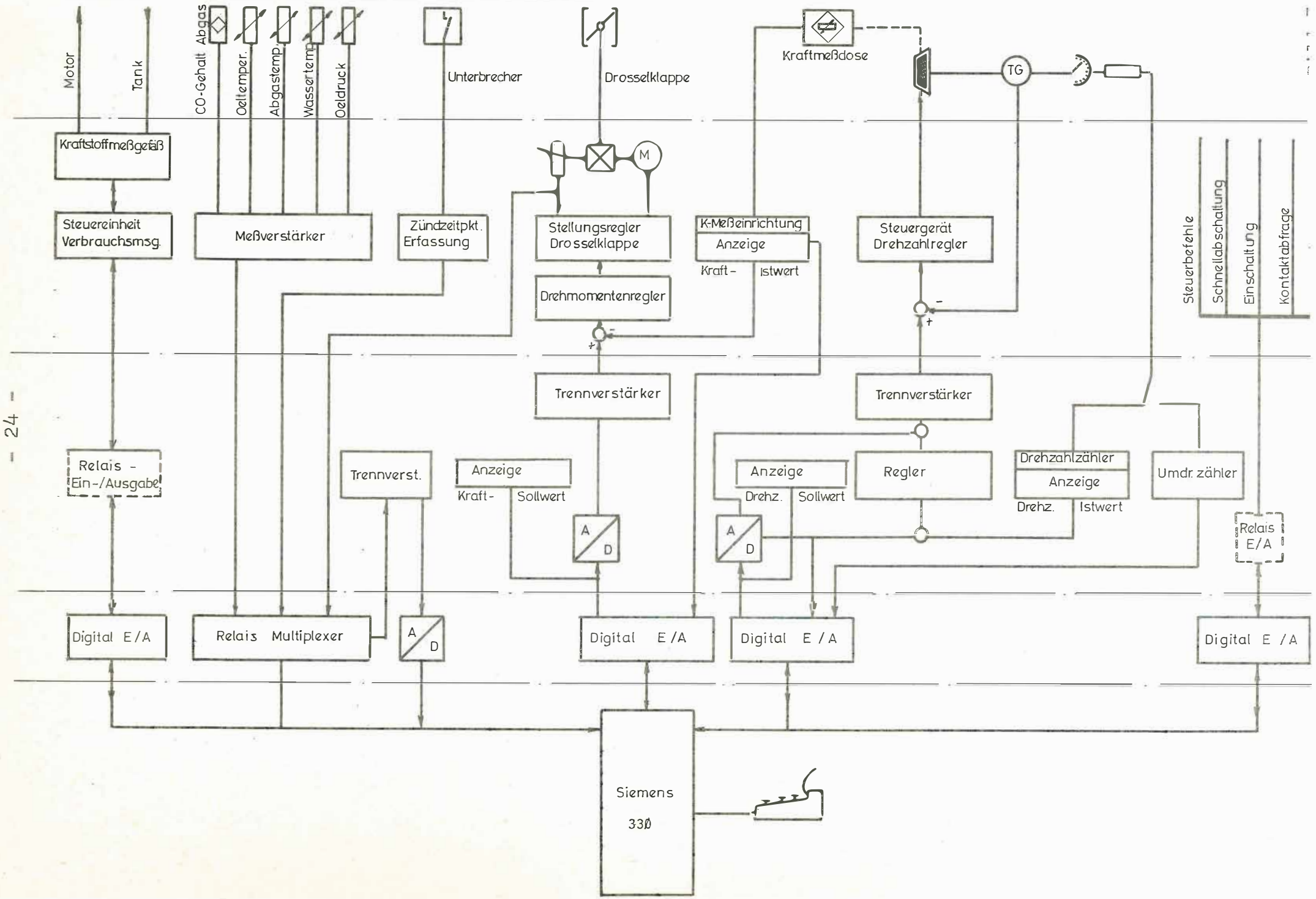
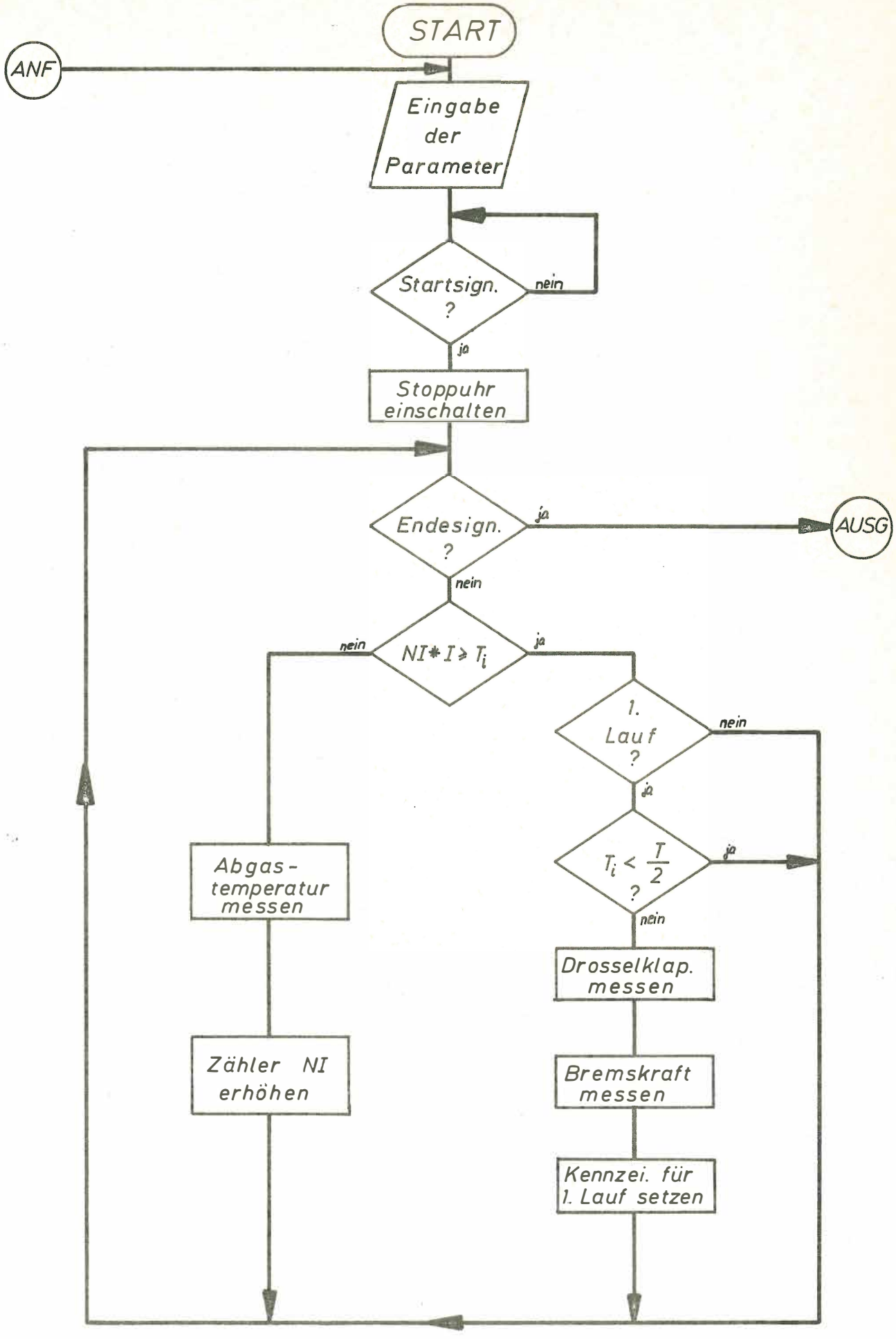
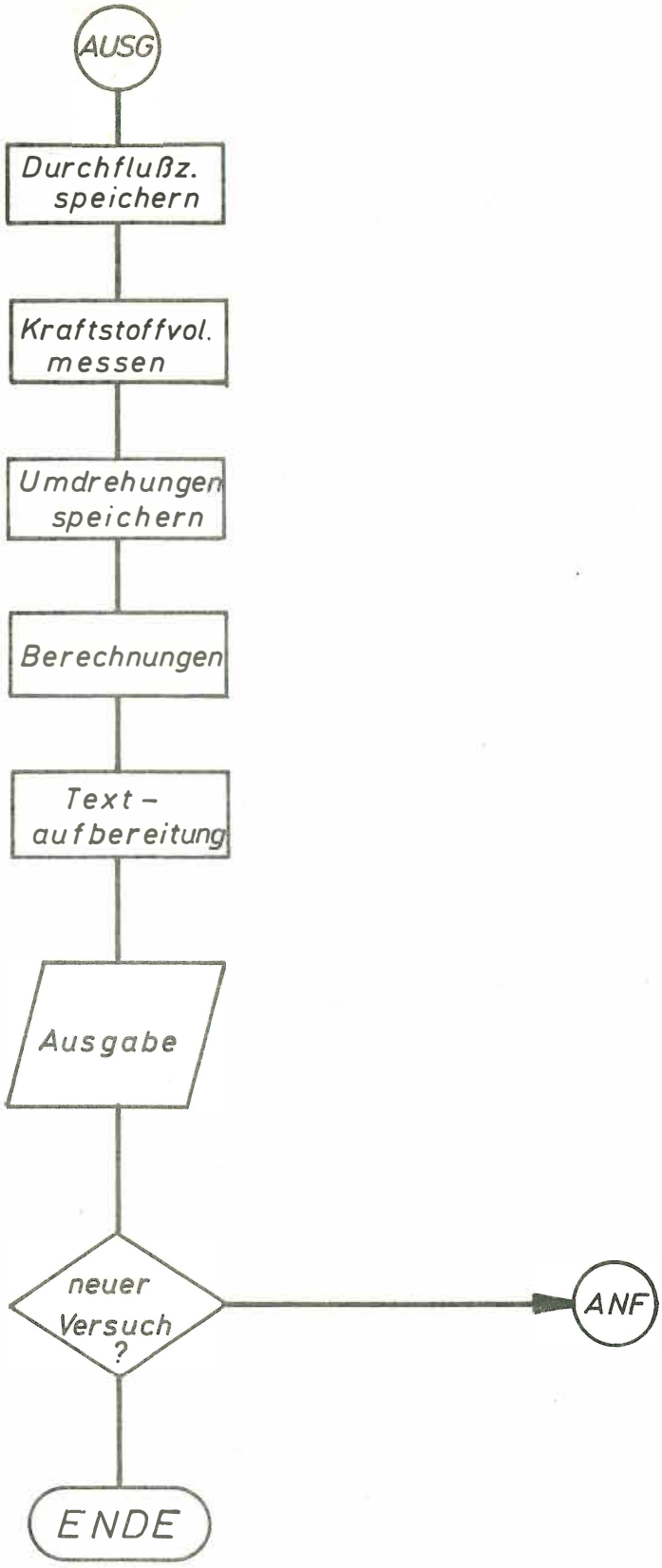

Betriebssystem
Systemsoftware
Anwendersoftware
E / A Blattschrb.
Platten-speicher
Lochkarten E / A
Lochstreif. E / A
Siemens Prozeßrechner 33Ø
Drucker
Plotter
Sichtgerät
CAMAC - Peripherie
Analog
Digital
Signalanpassung Meßwertaufbe-reitung
Kontroll- und Bedieneinrichtg.
Steuereinrichtg.
Regeleinrichtg.
Meßeinrichtungen
Prüfstand
Prüfling

Bild 2


Motor
Tank
CO-Gehalt Abgas
Oeltemper.
Abgastemp.
Wassertemp.
Oeldruck
Unterbrecher
Drosselklappe
Kraftmeßdose
TG
Kraftstoffmeßgefäß
Steuereinheit Verbrauchsmsg.
Meßverstärker
Zündzeitpkt. Erfassung
Stellungsregler Drosselklappe
M
K-Meßeinrichtung
Anzeige
Kraft-
Istwert
Steuergerät Drehzahlregler
Steuerbefehle
Schnellabschaltung
Einschaltung
Kontaktabfrage
Drehmomentenregler
Trennverstärker
Trennverstärker
Relais - Ein-/Ausgabe
Trennverst.
Anzeige
Kraft-
Sollwert
Anzeige
Drehz.
Sollwert
Regler
Drehzahlzähler
Anzeige
Drehz.
Istwert
Umdr. zähler
A
D
A
D
Relais E/A
Digital E/A
Relais Multiplexer
A
D
Digital E/A
Digital E/A
Digital E/A
Siemens
330


START
ANF
Eingabe der Parameter
Startsign. ?
nein
ja
Stoppuhr einschalten
Endesign. ?
ja
AUSG
nein
NI * I ≥ Ti
nein
ja
1. Lauf ?
nein
ja
Ti < T/2 ?
ja
nein
Abgas-temperatur messen
Drosselklap. messen
Zähler NI erhöhen
Bremskraft messen
Kennzei. für 1. Lauf setzen


AUSG
Durchflußz. speichern
Kraftstoffvol. messen
Umdrehungen speichern
Berechnungen
Text – aufbereitung
Ausgabe
neuer Versuch ?
ANF
ENDE

Datenerfassung und Datenauswertung mittels SIEMENS-301 am CERN Synchrozyklotron

H. Schroot, P. Skarek, N. Vogt-Nilsen.

Kurzfassung:

Ein SIEMENS-301 Doppelrechnersystem überwacht und registriert on-line einige hundert digitale Alarmsignale und etwa 40 Analogsignale, welche beim Betrieb des 600-MeV Protonenbeschleunigers von Bedeutung sind. Gewisse Betriebsdaten werden zusätzlich für spätere Abfrage und Auswertung - etwa über 3 Monate zurück - auf Platte gespeichert.

Das System soll zur Unterstützung der Fehleranalyse nach Betriebsstörungen und bei Experimenten zur Verbesserung des Beschleunigers verwendet werden.

Eine besonders disziplinierte Programmierung (Modularität, problemorientierte PROSA-Makros, absichtliche Beschränkung der Komplexität) ermöglichten den Aufbau und das Austesten eines umfangreichen Software-Systems von ca. 40 K PROSA (ca. 10 K PROSA-Makros) in der erstaunlich kurzen Zeit von 1 1/2 Mannjahren.

1. Einleitung.

Das CERN Synchrozyklotron (SC) beschleunigt Protonen auf eine Endenergie von 600 MeV. Wichtigste Untersysteme des Beschleunigers sind Ionenquelle, Vakuum, Magnetfeld (Führungsfeld), Hochfrequenz (Beschleunigungsfeld), Extraktionskanal und Teilchentransport zu den Experimenten, bzw. zu den Targets für Senkundärstahlen (Pionen, Müonen).

Die Zielsetzung bei der Entwicklung des zu beschreibenden Datenerfassungs- und Datenauswertungssystems (DAS) war folgende:

a) Hilfe bei der Fehleranalyse bei Betriebsstörungen und Betriebszusammenbrüchen, zusätzlich zum standardmäßig vorhandenen Überwachungs- und Alarmsystem,

b) Hilfe bei Beschleunigerstudien: schnellere Protokolle der Betriebsdaten, schnelle Rückgriffsmöglichkeit auf frühere Betriebsdaten.

2. Anlagenkonfiguration (Bild 1).

Es handelt sich um ein Doppelrechnersystem (zwei SIEMENS-301) mit - im Augenblick - nur sehr loser Kopplung über AKZ-Kanal 6[1], wobei Rechner 1 (DAS-1) digitale Alarmsignale verwaltet. Dies ist bereits an anderer Stelle ausführlich beschrieben[2].

Vorliegender Bericht behandelt fast ausschließlich das Programmsystem für Rechner 2 (DAS-2)[3,4].

3. Datenerfassung und Auswertung.

Rechner 2 registriert die Analogsignale aus den verschiedenen Untersystemen des Beschleunigers, prüft sie auf Grenzwertüberschreitung und speichert sie zur späteren Abfrage auf Platte.

Die Analogwerterfassung geschieht auf Wunsch (Tastendruck), zyklisch mit wählbarer Periode und durch DAS-1 über das Link getriggert, also bei Auftreten eines Alarms. Auf Platte werden nur die Rohwerte gespeichert. Wenn die Platte voll ist, wird automatisch die älteste Information überschrieben.

Die Analogwertauswertung geschieht auf Wunsch (Tastendruck) oder zyklisch mit wählbarer Periode. Bild 3 zeigt ein Protokoll bei aufgetretener Grenzwertüberschreitung, und das Protokoll einer Liste aller augenblicklichen Betriebsdaten ("Snapshot" genannt).

Bild 4 zeigt ein Protokoll eines Abfrageprogramms für momentane Betriebsdaten, und Bild 5 und 6 zeigen Resultate von Abfrageprogrammen, welche die auf Platte gepeicherte Information aufsuchen und ausdrucken.

Die Suchzeiten für binäres Suchen sind in der Größenordnung von maximal einigen Sekunden.

4. Software und Programmierung.

Es wäre zu bemerken, daß Systementwurf und Realisierung in der Hardware völlig konventionell sind. Wir denken aber, daß unsere software-mäßige Realisierung aus einigen Gründen Beachtung verdient.

Durch gezielte Selbstbeschränkung in der Komplexität, und durch diszipliniertes Programmierung in einer möglichst problemorientierten Makro-Sprache ist es uns gelungen, etwa 40 K PROSA (etwa 10 K Makro-Befehle) in einer erstaunlich kurzen Zeit fast fehlerfrei zu programmieren.

Bild 2 zeigt die Kernspeicherbelegung im DAS-2 Rechner. Obwohl Platte vorhanden, wurde nur ORG-A gewählt. Alle zeitkritischen Programme sind kernspeicherresident. Ein einziger Laufbereich wird durch ein Koordinator-Programm so verwaltet, daß mittels Makro MA ACTIVATE = NAME das Programm mit Namen "NAME" in den Laufbereich geladen und gestartet wird, falls dieser frei ist. Wenn dort gerade ein anderes Programm läuft oder wartet, wird NAME in eine Warteschlange eingetragen, welche jeweils beim Freiwerden des Laufbereichs abgearbeitet wird. Ein Rückschreiben von Programmen auf Platte findet nicht statt.

Die Verwendung von Makros wurde sehr weit getrieben, z.B. MA INCREM = A für A+1→A, MA TRANS = A,B für TEP A, TAS B, teilweise auch aus sprachlichen Gründen.

Andere Makros wie MA TEXT = AN = "TALLY PRINTER DOWN" statt üblicher BSAU/EXWA-Sequenzen erhöhen die Lesbarkeit und die Lokalität der Programme. Damit verbundener erhöhter Speicherplatzbedarf wurde bewußt in Kauf genommen und der bequemeren Programmierung der Vorzug gegeben.

5. Anwenderprogrammierung.

Abschließend sei noch erwähnt, daß wir beabsichtigen, Anwenderprogramme, welche über die erwähnten einfachen Abfrageprogramme hinausgehen, in FORTRAN IV zu schreiben.

Dazu muß man dem Anwender natürlich einige in FORTRAN aufrufbare Subroutinen und Funktionen zur Verfügung stellen, welche in einfacher und sicherer Weise Zugang zu den im Kernspeicher vorhandenen Momentanwerten des Betriebszustandes, und auf die auf der Platte vorhandenen früheren Daten Zugriff verschaffen[5,6].

6. Referenzen.

1) H. Schroot, "A medium speed program controlled Intercomputer Data-Link", PS-CD / 2.5 / 408, 16.2.1977 (Interner Bericht).

2) H. Beger, R. Cusack, A.Fiebig, H.Schroot, "The Data Acquisition System (DAS) for the improved CERN SC", Proc. 7th Conf. on Cyclotrons and their Applications (Birkhäuser, Basel, 1975), S 549-552.

3) P. Skarek, "What is DAS-2 ?", MSC-M-1, 21.1.1976 (Interner Bericht).

4) N. Vogt-Nilsen, "DAS-2 Users Guide", 1977 (Interner Bericht).

5) P. Skarek, "Note on the User's Interface to the DAS-2 Database", PS-CDI/98, 9.4.1976 (Interner Bericht).

6) P. Skarek, "An Example of a DAS-2 Application Program written in FORTRAN", PS-CD/290, 20.10.1976 (Interner Bericht).

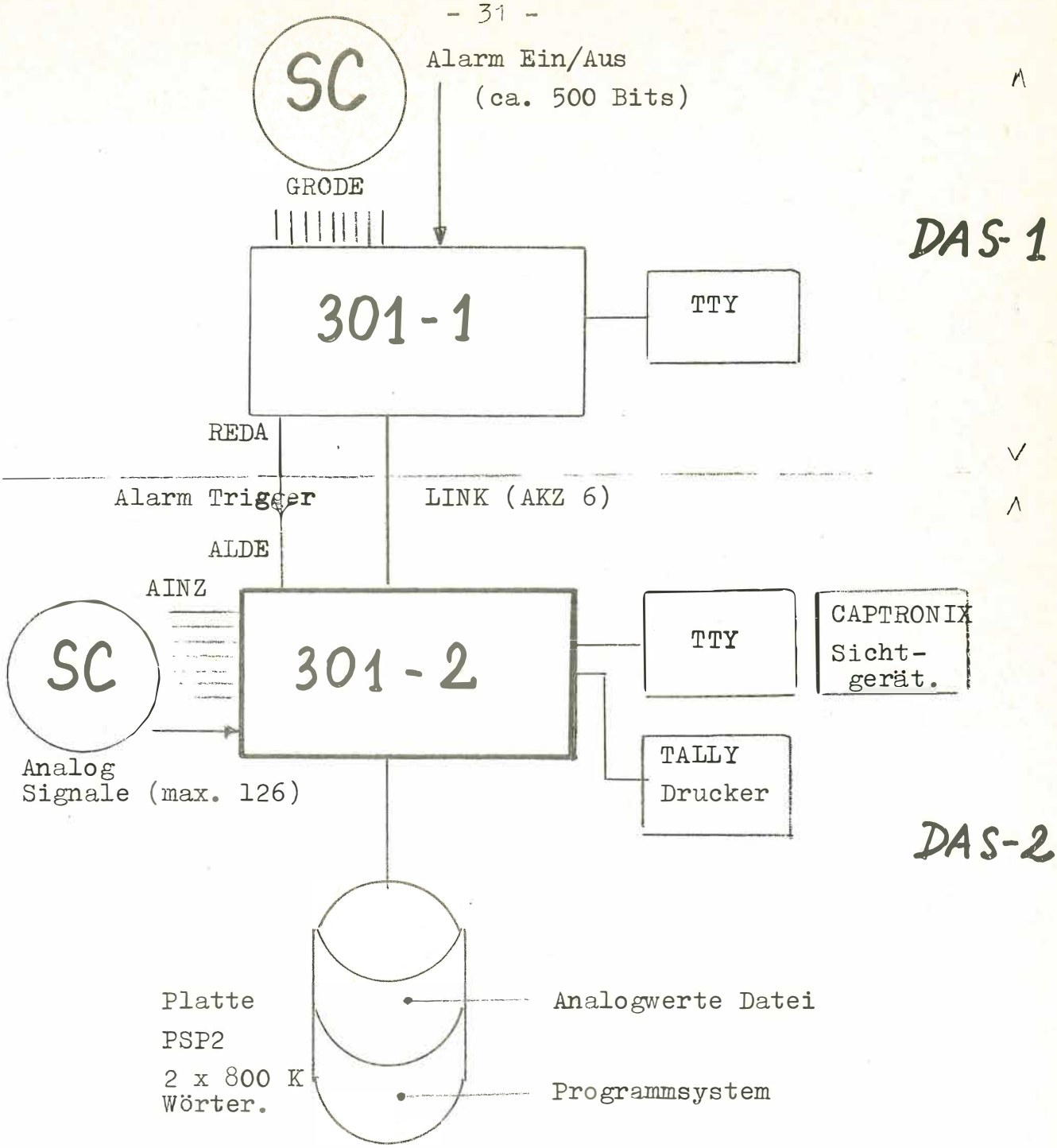


Bild 1: Anlagenkonfiguration.

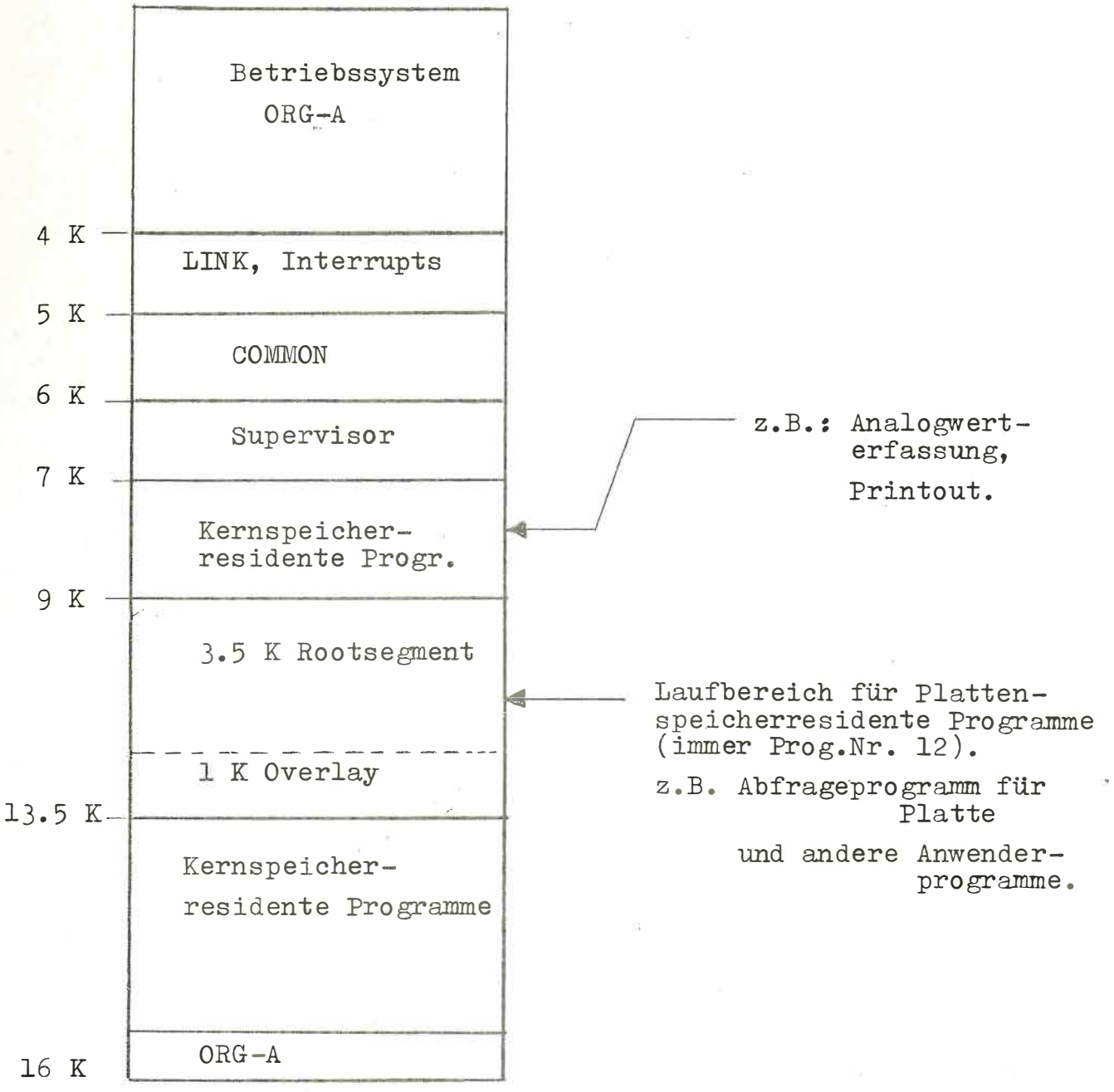


Bild 2: Speicherplatzbelegung.

Kaltstart

```
 * DAS-2 COLD START * VS. 1.6
   ---------------
DAS-2: DASD POSITIONED AT END OF BLOCK    7950  / FREE
DAS-2: STANDARD INSTALLATION
       BSAU FOR LIMITS CHECK      0
       ANALOG SCAN PERIOD       10 MIN
       SNAPSHOT    PERIOD        1 H

  DAS-2: DATA/TEXT AREA LOADED /   TYPE NO.   1751 /   41 SIGNALS
  DAS-2: PRT1/VS 2.3
  DAS-2: ANALOG SCAN PROGRAM ASCN LOADED
  DAS-2: SVIS /VS 2.4
  DAS-2: LINK ENABLE / SCAN START BY LINK IS ON
  DAS-2: COORDINATOR VS.3.9
  COLD START END
```

15.06.76 / 10.48.18

Grenzwertkontrolle

Ausdruck auf Blattschreiber oder TALLY-Drucker

```
*SCAN*
  DAS-2: ASCN VS. 3.3

   10.48.37:   I(EMC)      EMC CURRENT         LOWER LIMIT
   10.48.37:   R/FORVAC    ROTCO FOREVACUUM    LOWER LIMIT
   10.48.37:   R/ENDVAC    ROTCO ENDSH. VAC    LOWER LIMIT
   10.48.37:   R/TVPVAC    TVP 900 VACUUM      LOWER LIMIT
   10.48.37:   R/VAC       ROTCO MAIN VAC.     LOWER LIMIT

CHANGE;;
*SCAN*
   10.53.16:   I(EMC)      EMC CURRENT         LOWER LIMIT
   10.53.16:   R/FORVAC    ROTCO FOREVACUUM    LOWER LIMIT
   10.53.16:   R/ENDVAC    ROTCO ENDSH. VAC    LOWER LIMIT
   10.53.16:   R/TVPVAC    TVP 900 VACUUM      LOWER LIMIT
   10.53.16:   R/VAC       ROTCO MAIN VAC.     LOWER LIMIT
```

"Snapshot" - Momentanzustand des Beschleunigers.

```
DAS-2: SNAPSHOT/VS 2.2      15.06.76 /  10.50.10

15.06.76 /  10.50.10

  TIME      A N A L O G   S I G N A L        VALUE          LOWER LIMIT     UPPER LIMIT
                                                                          LIMIT CHECK ON
10.49.33   I(SC)       SC MAGN. CURRENT   -1810.      A      -1850.          1850.
10.49.33   I(EMC)      EMC CURRENT            0.      KA        10.00          12.10    OUT OF LIMITS
10.49.33   R/VIBR      ROTCO VIBROGUARD       1.1     MMM        0.            25.
10.49.33   I(LA1)      LA1 CURRENT            0.      A       -140.0          140.0
10.49.33   I(LA2)      LA2 CURRENT            0.      A       -140.0          140.0
10.49.33   I(LA3)      LA3 CURRENT            4.875   A       -140.0          140.0
10.49.33   EMCL        EMC: LEFT POS.         2.6     VOLT       0.           10.
10.49.33   EMCR        EMC: RIGHT POS.        3.5     VOLT       0.           10
10.49.33   DCA         DEE CENTR. AMPL.      19.5     KV        15.0          30.0
10.49.33   CPA         CEE PEAK AMPL.        12.6     KV         5.00         16.0
10.49.33   R/SPEED     SC-ROTCO SPEED       364.      C/S      200.           500.
10.49.33   R/FORVAC    ROTCO FOREVACUUM      -0.2     MV         0.           10.      OUT OF LIMITS
10.49.33   R/ENDVAC    ROTCO ENDSH. VAC      -0.36    T-6        0.           10.      OUT OF LIMITS
10.49.33   R/TVPVAC    TVP 900 VACUUM        -0.22    T-5        0.           10.      OUT OF LIMITS
10.49.33   R/VAC       ROTCO MAIN VAC.       -2.0     T-6        0.           10.      OUT OF LIMITS
10.49.33   ********    ***** TEST *****       4.93125 VOLT    -10.0000         9.99999
```

Bild 3: Protokolle vom Kaltstart, Grenzwertkontrolle und "Snapshot".

```
QUIZ;;
  DAS-2: QUIZ/VS 1.1      15.06.76 /  10.52.02

QUIZ: TELE;;

I(SC);

  10.49.33   I(SC)          -1810.      A
```

Ausgabe eines Wertes auf Blattschreiber.

```
QUIZ:
**LINK** DAS-1 ALARM
TELE;;

DCA CPA R/SPEED R/VAC;

  10.53.16   DCA             19.5      KV
  10.53.16   CPA             12.6      KV
  10.53.16   R/SPEED        364.       C/S
  10.53.16   R/VAC           -1.9      T-6    OUT OF LIMITS
```

Ausgabe mehrerer Werte auf Blattschreiber.

```
QUIZ: TALLY;;

I(LA3)  EMCL  EMCR  DCA  CPA  ERREUR;

QUIZ: LOGOUT;;

PROGRAM END
```

Ausgabe mehrerer Werte, inklusive Grenzwerte auf TALLY-Drucker.

```
.06.76 /  10.52.02

TIME      A N A L O G   S I G N A L        VALUE          LOWER LIMIT     UPPER LIMIT
-------------------------------------------------------------------------------------

).53.16   I(LA3)    LA3 CURRENT            5.000   A       -140.0          140.0
).53.16   EMCL      EMC: LEFT POS.          2.6    VOLT      0.             10.
).53.16   EMCR      EMC: RIGHT POS.         3.5    VOLT      0.             10.
).53.16   DCA       DEE CENTR. AMPL.       19.5    KV       15.0            30.0
).53.16   CPA       CEE PEAK AMPL.         12.6    KV        5.00           16.0
).53.16   ERREUR          UNKNOWN SIGNAL
```

Bild 4: Ausdrucke vom Programm QUIZ
(protokolliert momentane Betriebsdaten, die Anzahl ist wählbar).

```
RETR;;                                   Suchprogramm für Analogwert-
                                                          datei.
DAS-2: RETR / VS. 2.1
        PROTOTYPE RETRIEVAL PROGRAM

..FIND,15.06.76/11.00;;  <-------------- gewünschtes Datum / Zeit.
SEARCH FOR     DATE:  15.06.76    TIME:  11.00.00
NEAREST FOUND DATE:  15.06.76    TIME:  10.49.33

..FULL;;  <------------------------------ erzeugt untenstehendes Protokoll
                                              am TALLY-Drucker.

..LOGOUT;;
  RETR - END
```

TIME 15.06.76 / 10.49.33

ANALOG SIGNAL		VALUE		LOWER LIMIT	UPPER LIMIT	LIMIT CHECK ON
I(SC)	SC MAGN. CURRENT	-1810.	A	-1850.	1850.	
I(EMC)	EMC CURRENT	0.	KA	10.00	12.10	OUT OF LIMITS
R/VIBR	ROTCO VIBROGUARD	1.1	MMM	0.	25.	
I(LA1)	LA1 CURRENT	0.	A	-140.0	140.0	
I(LA2)	LA2 CURRENT	0.	A	-140.0	140.0	
I(LA3)	LA3 CURRENT	4.875	A	-140.0	140.0	
EMCL	EMC: LEFT POS.	2.6	VOLT	0.	10.	
EMCR	EMC: RIGHT POS.	3.5	VOLT	0.	10.	
DCA	DEE CENTR. AMPL.	19.5	KV	15.0	30.0	
CPA	CEE PEAK AMPL.	12.6	KV	5.00	16.0	
R/SPEED	SC-ROTCO SPEED	364.	C/S	200.	500.	
R/FORVAC	ROTCO FOREVACUUM	-8.2	MV	0.	10.	OUT OF LIMITS
R/ENDVAC	ROTCO ENDSH. VAC	-0.36	T-6	0.	10.	OUT OF LIMITS
R/TVPVAC	TVP 900 VACUUM	-0.22	T-5	0.	10.	OUT OF LIMITS
R/VAC	ROTCO MAIN VAC.	-2.0	T-6	0.	10.	OUT OF LIMITS
********	***** TEST *****	4.93125	VOLT	-10.0000	9.99999	

Bild 5: Plattensuchprogramm RETR - protokolliert gesamte Betreibsdaten für beliebiges Datum/Zeit.

```
QUEST;;
  DAS-2: QUEST/VS 1.5     28.01.77 / 09.24.04

QUEST: HOUR;;
FROM  25 1 12  TO  27 1 20;

QUEST: TTY;;
I(EMC)  EMCL  EMCR;
```

Suchprogramm für Analogwert-datei.

gewünschter Abstand zur nächsten Analogwerterfassung auf Platte

gewünschter Zeitbereich für das Protokoll.

ausgewählte Analogsignale.

DATE/TIME	DAS-1 ALARM	I(EMC) KA	EMCL MM	EMCR MM
DATE	25.01.77			
TIME	10.30.31.2	1.011	9.93	9.98
DATE	26.01.77			
TIME	14.32.50.1	0.	9.93	9.98
	15.04.01.3 ***	11.29	9.92	9.98
	16.03.07.0	11.30	9.92	9.98
	17.06.01.3	11.30	9.93	9.98
	18.06.01.3	11.39	10.5	9.85
	19.06.01.3	11.39	10.5	9.85
	20.06.01.3	11.30	10.0	9.85
	21.06.01.3	11.29	10.0	9.85
	22.06.01.3	11.29	10.0	9.85
	23.06.01.3	11.30	10.0	9.85
DATE	27.01.77			
TIME	00.06.01.3	11.30	10.0	9.85
	01.02.11.9	0.	10.0	9.85
	02.06.01.3	0.	10.0	9.85
	03.06.01.3	0.	10.0	9.85
	04.06.01.3	0.	10.0	9.85
	05.06.01.3	0.	10.0	9.85
	06.06.01.3	0.	10.0	9.85
	07.06.01.3	0.	10.0	9.85
	08.06.01.3	0.	10.0	9.85
	09.05.00.1	0.	10.0	9.85
	10.04.01.3 ***	0.	10.0	9.85
	11.02.48.0 ***	0.	10.0	9.85
	12.04.01.3	0.	10.0	9.85
	13.04.01.3	0.	10.0	9.85
	14.04.01.3	0.	10.0	9.85
	15.02.17.3	0.	10.0	9.85
	16.02.01.3 ***	11.30	10.0	9.85
	17.04.07.6	11.30	10.0	9.85
	18.06.01.3	11.29	10.0	9.85
	19.06.01.3	11.30	10.0	9.85
	20.06.01.3	11.30	10.0	9.85

```
QUEST: LOGOUT;;

PROGRAM END
```

Bild 6: Plattensuchprogramm für detaillierte Anfragen.

DER PROZESSRECHNER SIEMENS 310S ALS TELEFONISCHES FAHRPLANAUSKUNFT-TERMINAL MIT INDIVIDUELLEM BENUTZERDIALOG

FRIEDRICHSHAFEN, 15.4.1977
HOCHMUTH, ABT. EER

Einleitung

Unter der Bezeichnung

DAS TELEFON ALS TERMINAL FÜR RECHNERGESTEUERTE INFORMATIONSSYSTEME

wurde im Auftrag des Bundesministeriums für Forschung und Technologie (BMFT) die Realisierung neuer rechnergesteuerter Telekommunikationsformen im bestehenden Fernsprechnetz untersucht (Forschungsbericht BMFT-NT 636).

Die in einer Studie zusammengefaßte Untersuchung zeigt Wege, wie jedem Fernsprechteilnehmer der Zugriff auf neuartige Informationsdienste mit Sprach- und Bildausgabe zum Abruf individueller Informationen ermöglicht werden kann.

INFORMATIONSSYSTEME MIT SPRACHAUSGABE

Die für die Teilnehmerseite einfachste Form eines Informationssystems bedient sich der Sprache als Ausgabemedium, bekannt in Form von Zeitansage, Wetterbericht, Küchenrezepte usw. Solche Systeme geben im allgemeinen feste, analog gespeicherte Texte an einen Anrufer weiter.

Für umfassende, individuelle, aktuelle und daher variable Textausgaben ist eine rechnergesteuerte synthetische Spracherzeugung Voraussetzung. Im Dialog mit der Auskunfts-Zentrale (Dateneingabe über Wählscheibe/Wähltastatur) kann der Teilnehmer spezielle Detailinformationen abrufen; Abb. 1.

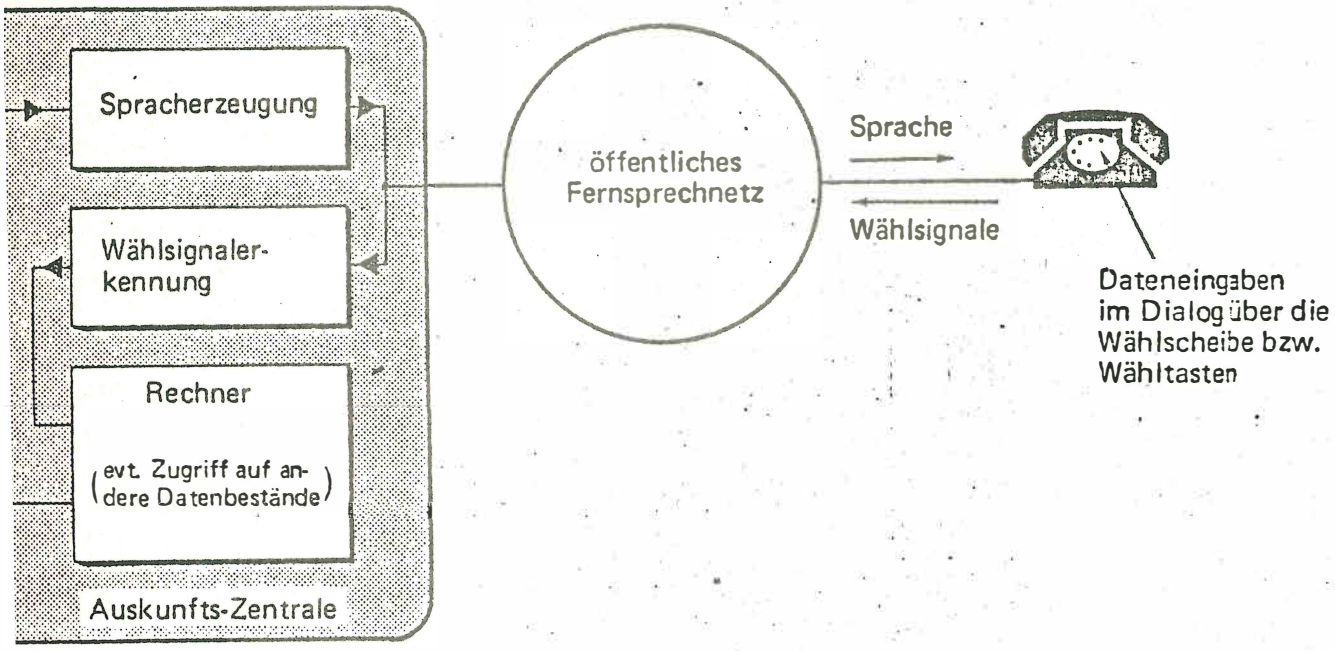


Abb. 1: Informationssystem mit Sprachausgabe

INFORMATIONSSYSTEME MIT BILDAUSGABE

Zur Auswahl und zum Vergleich von alternativen Vorschlägen (z.B. mehrere gleichwertige Zugverbindungen) ist eine schriftliche oder grafische Ausgabe der Auskünfte besser geeignet. Durch die Nutzung des Heimfernsehers ist eine solche Informationsdarstellung auch für private Haushalte kostengünstig realisierbar. Hierbei wird das Fernsehgerät über ein Zusatzgerät an das Telefon und damit an die Auskunftzentrale angeschlossen; Abb. 2.

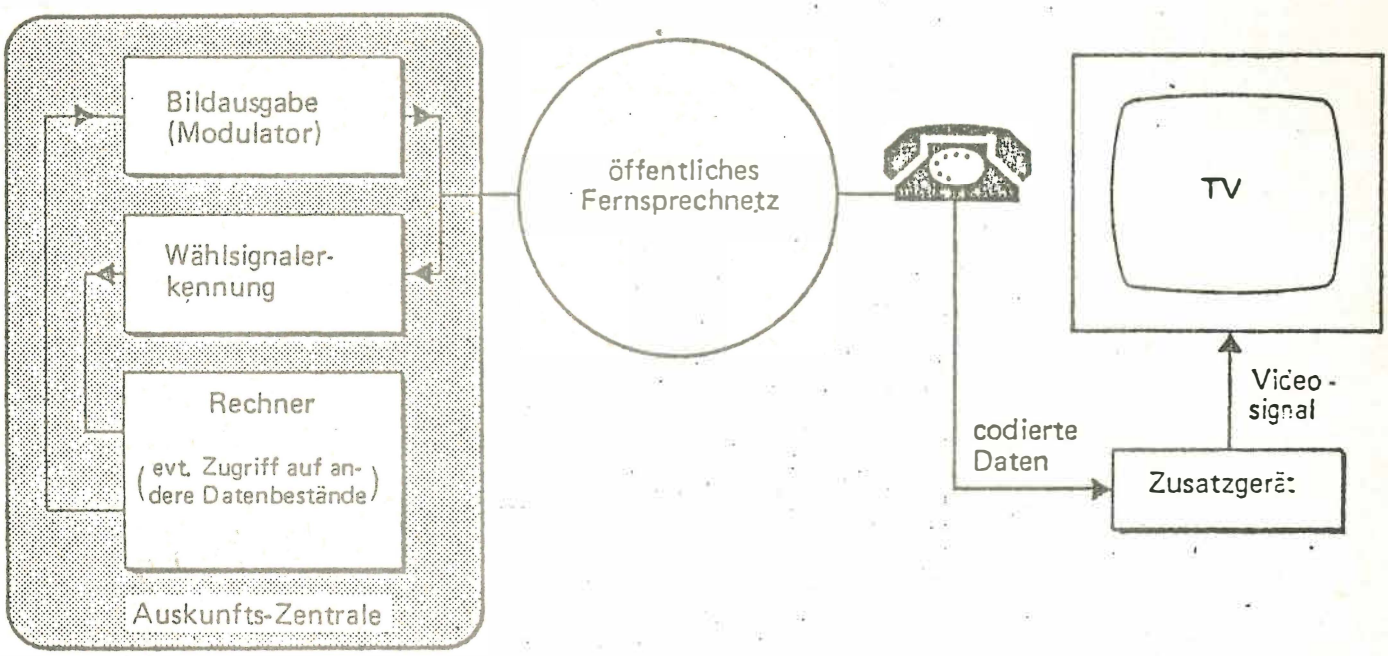


Abb. 2: Informationssystem mit Bildausgabe

Abb. 3 zeigt den internen Aufbau des Zusatzgerätes das folgende Funktionen erfüllt:

- Auskopplung der über das Fernsprechnetz übertragenen Information auf einfache Weise ohne Änderung am Fernsprechhauptanschluß des Teilnehmers, realisiert durch Anschluß an den Zweithöreranschluß des Fernsprechgerätes.

- Aufbereitung der Daten und Speicherung der kompletten Information eines Bildes, auch nach Abbruch der Telefonverbindung.

- Erzeugung der darzustellenden Symbole, Video-Signal-Erzeugung und Einkopplung des Fernsehsignals auf einfache Weise in den Fernsehempfänger.

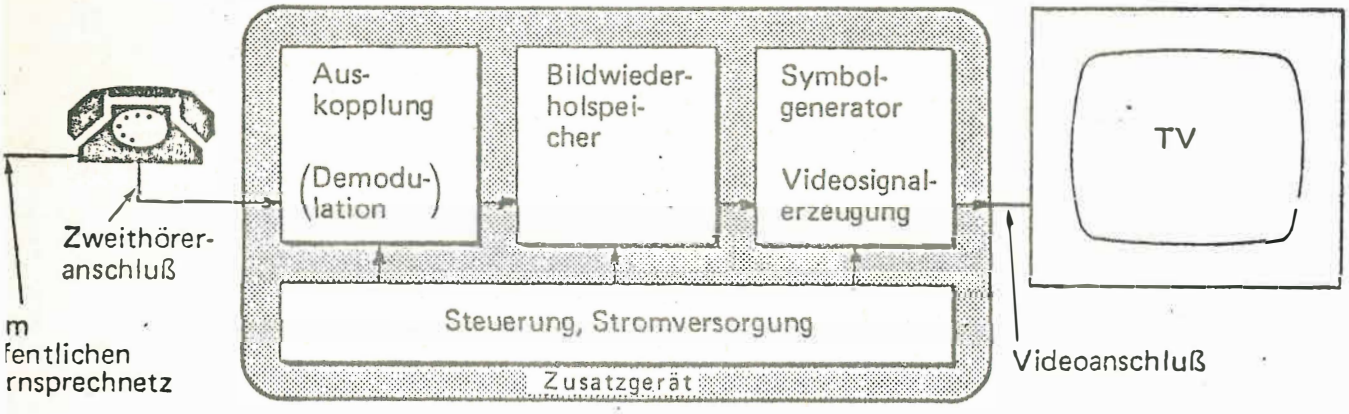


Abb. 3: Zusatzeinrichtung zur Kopplung von Telefon und Heimfernseher

Das darzustellende Bild wird aus bitseriell übertragenen Schriftzeichen und Grafiksymbolen und mit Hilfe von Steuerzeichen aufgebaut. Im Bildwiederholspeicher ist das Bild in codierter Form gespeichert. Daraus wird es mit Hilfe des Symbolgenerators und mit entsprechenden Schaltungen in ein Videosignal umgewandelt und dem Fernsehgerät in Form von 2 identischen Halbbildern zugeführt. Der Bildaufbau und die Organisation des Bildspeichers ist zeichenplatzorientiert. Dargestellt werden können 128 Schrift-

zeichen (ASCII-Zeichen, Kleinbuchstaben und Sonder-symbole) und 64 Grafiksymbole in 7 verschiedenen Farben und einem Blinkzustand.

Für die Erprobung und Beurteilung solcher neuer Kommunikationsformen wird zur Zeit das Pilotsystem "Fahrplanauskunft für abfahrende Züge ab Frankfurt" realisiert (kurz Fahrplanauskunft Frankfurt = FAF).

PILOTSYSTEM FAHRPLANAUSKUNFT FRANKFURT SYSTEMBESCHREIBUNG (SIEHE ABBILDUNG 4)

Dieses Pilotsystem verwendet Datenbestände der Deutschen Bundesbahn, die für die bereits im Betrieb befindliche Elektronische Reisezugauskunft (ERA) der DB erstellt wurden und immer dem aktuellen Fahrplan entsprechen. Zur Zeit sind ab Frankfurt zu etwa 300 Zielorten alle Zugverbindungen abrufbar gespeichert. Der Auskunftsuchende bedient das Auskunftsystem über das öffentliche Fernsprechnetz. Für das Auskunftsystem ist sowohl eine Sprach- als auch eine Bildausgabe vorgesehen. Für die Sprachausgabe werden keinerlei Zusatzgeräte beim Fernsprechteilnehmer benötigt, so daß alle Fernsprechteilnehmer im Ortsnetz von Frankfurt nach Einrichtung der Auskunftzentrale und Inbetriebnahme des Systems (2.Jahreshälfte 1977) als Benutzer in Betracht kommen.

Die Bildausgabe, die vorläufig nur im kleineren Rahmen erprobt werden soll, benötigt beim Teilnehmer ein Zusatzgerät zur Kopplung von Telefon und Fernsehgerät über den Zweithöreranschluß.

FAHRPLANAUSKUNFT
SYSTEMÜBERSICHT

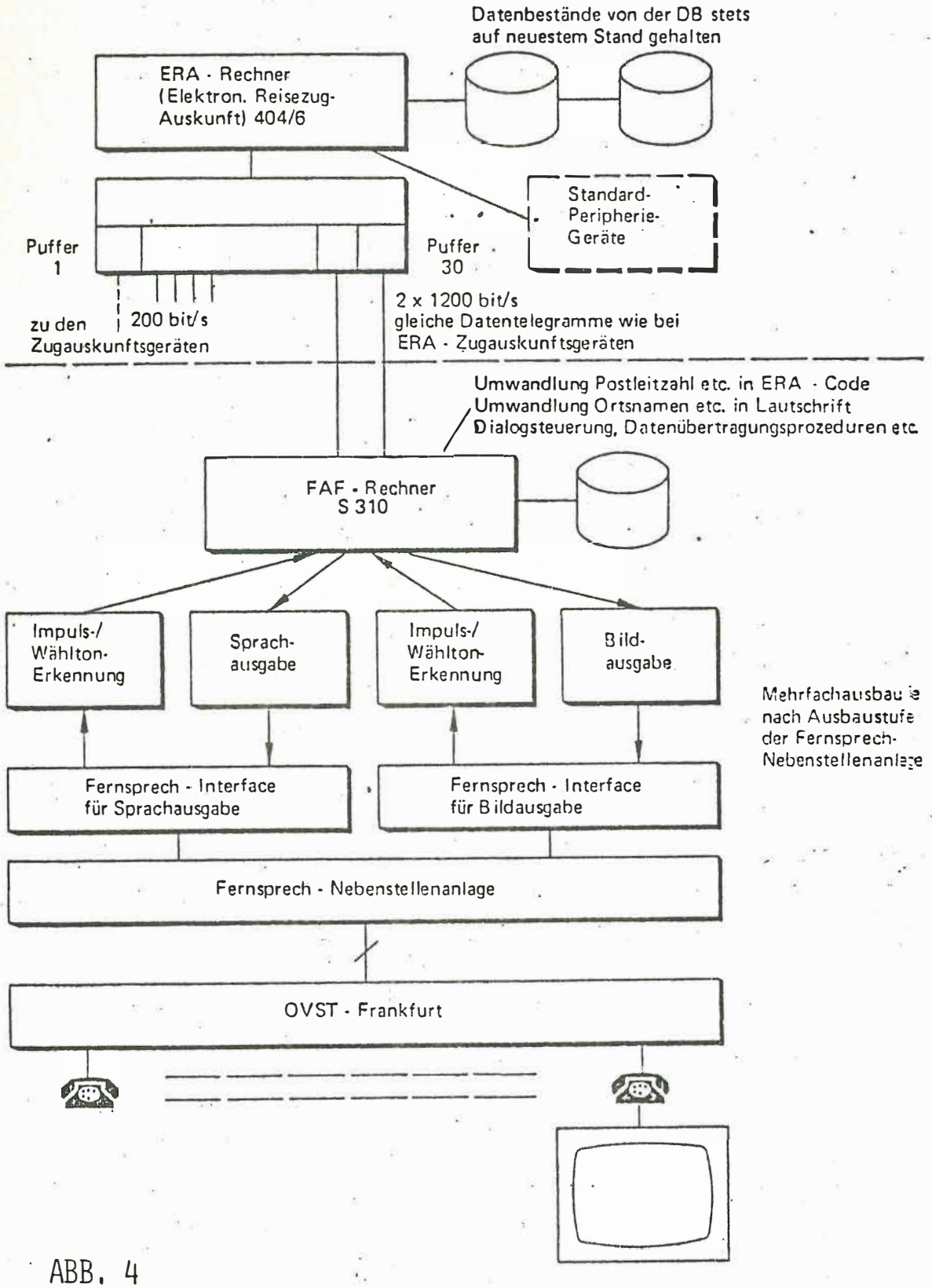


ABB. 4

GERÄTEKONFIGURATION DER AUSKUNFTSZENTRALE (SIEHE ABB. 5)

Die Auskunftszentrale besteht aus einem Prozeßrechner Siemens 310S mit 32K-Kernspeicher, einem Plattenspeicher 3941, einem Bedienblattschreiber, einer in CAMAC-Norm aus Standardkomponenten erstellten Rechnerperipherie und einer Fernsprech-Nebenstellenanlage, die über Durchwahleinrichtungen mit mehreren Amtsleitungen an das öffentliche Fernsprechnetz von Frankfurt angeschlossen ist. Jeder Amtsleitung ist ein CAMAC-Modul zugeordnet zur Steuerung der Fernsprech-Nebenstellenanlage, zur Erkennung der Nachwahlimpulse, die vom Anrufer zur Dateneingabe abgegeben werden, zur Einkopplung der Sprachausgabe in das Fernsprechnetz und zur Modulation und Einkopplung der Bildausgabe in das Fernsprechnetz. Für die Ankopplung des FAF-Rechners an den ERA-Rechner dient ein CAMAC-Datenfernübertragungsmodul (DFUA) mit V-24-Schnittstelle. Ein weiterer CAMAC-Modul führt die aktuelle Uhrzeit mit Datum.

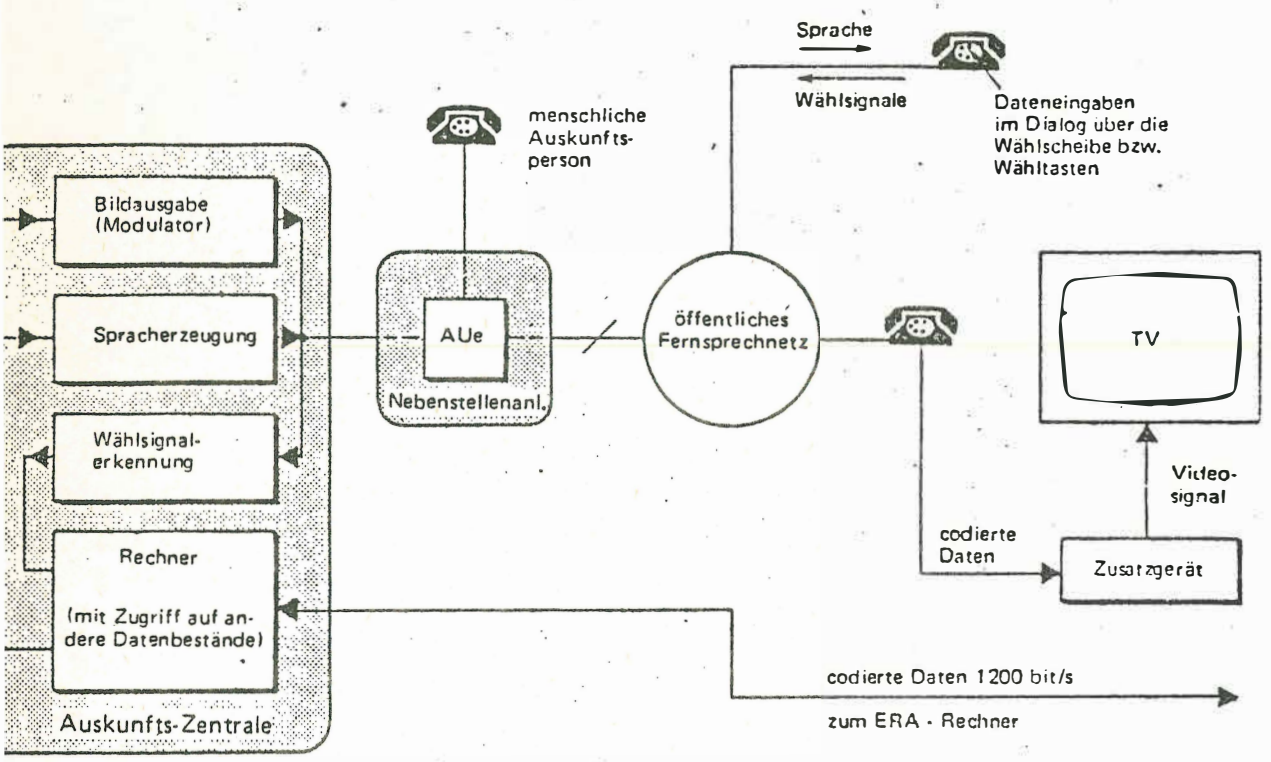


Abb.5: Auskunftzentrale FAF

Als Sprachausgabegerät kommen sowohl Geräte der Firma VOTRAX als auch Entwicklungen der Deutschen Bundespost (Forschungs-Institut der DBP beim Fernmeldetechnischen Zentralamt) zum Einsatz.

Der Modulator für die Bildausgabe ist Bestandteil eines CAMAC-Moduls. Die Daten werden bitseriell mit Hilfe von Frequenzumtastung in binär modulierter Form übertragen. Die Datenübertragung erfolgt asynchron mit Start- und Stopbit nach DIN 66022.

STEUERUNG DES AUSKUNFTSYSTEMS

Die Software des FAF-Rechners realisiert den Auskunft-Dialog mit mehreren Anrufern gleichzeitig. Der Rechner gibt zu Beginn einer Auskunft einen Meldetext, mit dem sich das System vorstellt und Bedienungshinweise an den Anrufer aus. In einzelnen Dialogschritten werden die für eine Auskunfterteilung benötigten Parameter angefordert und durch entsprechende Rückmeldungen an den Anrufer quittiert. Der Anrufer hat damit die Möglichkeit, Eingaben auf ihre Richtigkeit zu überprüfen und im Fehlerfall neu einzugeben. Außerdem werden durch entsprechende Eingaben mehrere Zugverbindungen zu einem Zielort mitgeteilt, ohne daß die einzelnen Parameter für die Zugverbindung neu eingegeben werden müssen. Geübte Benutzer können den Dialogablauf verkürzen, indem sie alle für eine Fahrplanauskunft benötigten Parameter, ohne die jeweilige Aufforderung abzuwarten, nacheinander eingeben. Wenn Anrufer mit dem Auskunftsystem nicht zurechtkommen oder wenn die gewünschte Auskunft nicht erteilt werden kann, wird automatisch oder auf Wunsch des Anrufers an eine menschliche Auskunft weitervermittelt.

Für die Nennung seines Fahrzieles gibt der Anrufer die Postleitzahl oder die Ortskennzahl des Zielortes und die gewünschte Abfahrtzeit ein. Aus diesen Angaben stellt der Rechner ein Datentelegramm an die Elektronische Reisezugauskunft der DB (ERA) zusammen und sendet es an den ERA-Rechner. Der ERA-Rechner sendet ein Datentelegramm mit einem Fahrplanauszug für den angewählten Zeitraum zu dem gewünschten Zielort. Daraus wird im FAF-Rechner die geeignetste Zugverbindung ausgewählt und dem Anrufer in gesprochener Form (synthetische Sprache) oder in codierter Form für eine entsprechende Bildausgabe ausgegeben.

Falls die günstigste Verbindung Züge enthält, die nicht an jedem Reisetag verkehren oder nicht beide Wagenklassen führen, sind weitere Dialogschritte erforderlich, in denen der Anrufer den Reisetag und die gewünschte Wagenklasse angibt.

Die einzelnen Sprechtexte, aus denen die Ausgaben an den Anrufer zusammengestellt werden, sind in phonetischem Code auf dem Plattenspeicher abgelegt. Außerdem enthält der Plattenspeicher Dateien über die Zuordnung von Postleitzahl, Ortskennzahl, Ortsnamen und ERA-Nummer sowie Dateien mit Formaten für die Bildausgabe und Tabellen für Code-Umwandlung.

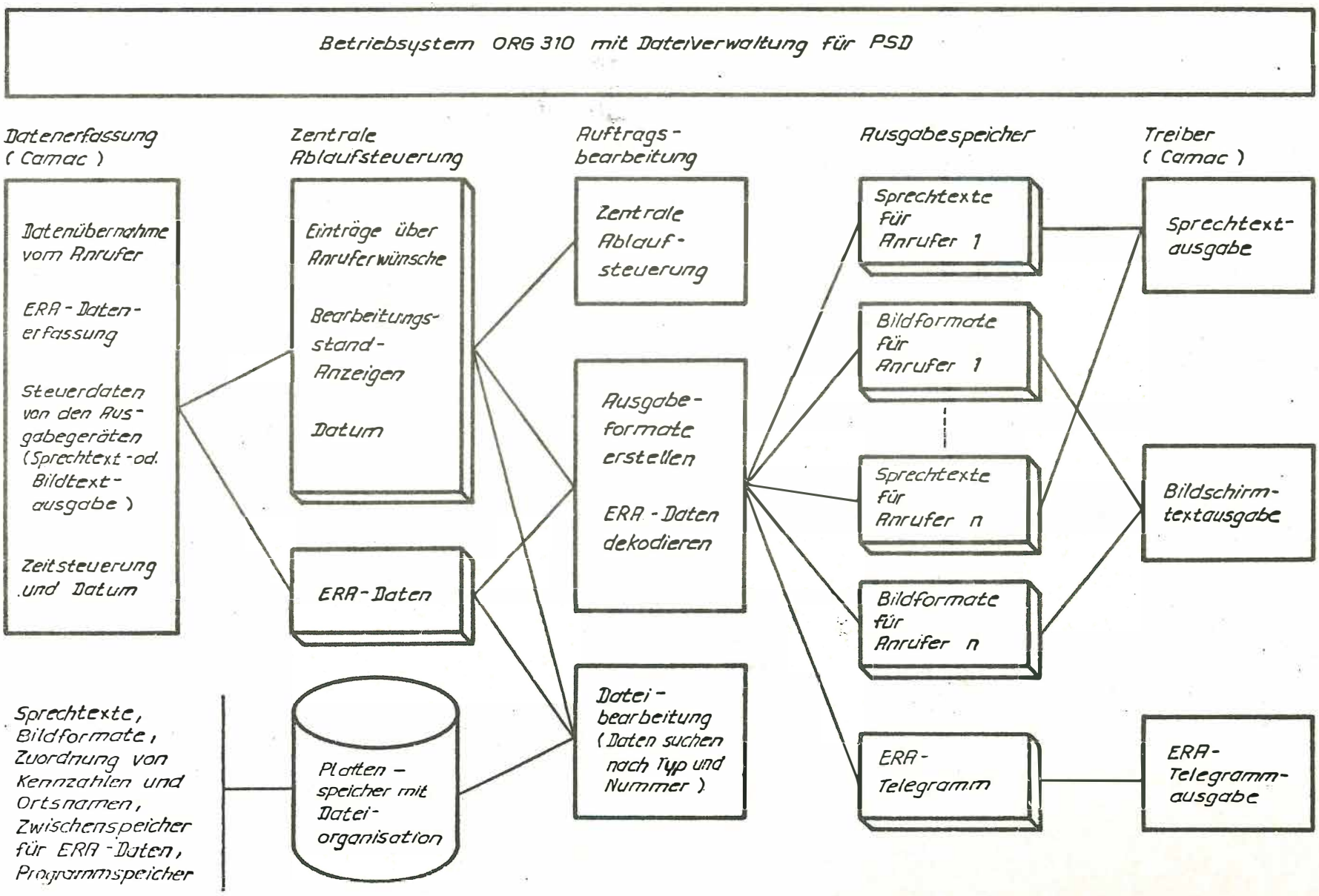
SOFTWARE DES FAF-RECHNERS

Die gesamte Software des FAF-Rechners wird auf einem Prozeßrechner Siemens 330 mit leistungsfähigen Ein-/Ausgabegeräten erstellt und als Grundsprachemodule in den FAF-Rechner geladen. Das Betriebssystem wird aus der MOBI 310 generiert und in Urladeformat ausgegeben.

Die Software für die Fahrplanauskunft wird auf dem Plattenspeicher des FAF-Rechners urladefähig abgelegt.

Außer dem Programmsystem für die Abwicklung der Fahrplanauskunft werden noch Programme zum Vorbereiten und Modifizieren der Plattenspeicher-Datenbestände und zum Erstellen von Sprechtexten entwickelt.

Struktur der FAF- Software



Karl-Friedrich Bamberger
SIEMENS AG - E STE 33

Karlsruhe, 21.4.1977

Maskensoftware MASK 310

1. Einleitung

Die Maskensoftware MASK 310 ist ein Programmiersystem zur Datenerfassung über Zeichenbildschirmeinheiten.
Bei der Kommunikation mit dem Rechner über Zeichenbildschirmeinheiten mittels Masken erfolgt z.B. die Parametervorgabe nicht mit der bisher üblichen Kommandotechnik sondern über Masken.

Hierzu legt ein Programm dem Benutzer quasi in Form eines Fragebogens oder Formulars eine Maske vor und der Benutzer trägt in bestimmte Felder die variablen Parameter ein, wobei ihn das Programm durch den Sichtgerätecursor von Feld zu Feld führt.
Die zeichenweise Eingabe wird vom Programm einer Prüfung unterzogen. Ist eine Eingabe im Sinne der in der Maske festgelegten Prüfbedingungen unzulässig, so bleibt der Cursor stehen oder wird gar zurückpositioniert. Daraufhin muß diese Eingabe wiederholt werden. Eingaben über Masken sind damit entsprechend der individuell vorgebbaren Prüfvorschriften fehlerfrei.
Da die Masken selbsterklärend sind, entfällt in der Regel das Nachlesen von Hantierungsvorschriften. Probleme, wie man sie hin und wieder mit einer Kommandosyntax hat, entfallen ganz.

2. Komponenten des Programmsystems MASK 300

Nach dieser kurzen Darstellung der wesentlichen Merkmale von Masken beim rechnergesteuerten Datenerfassungsdialog werden die Hauptkomponenten der Maskensoftware 310 (Bild 1) etwas näher betrachtet. Dies sind:

- Maskencompiler inklusive Beschreibungssprache f.Bildschirmmasken
- Maskenverwaltung auf PSD

} off-line Komponenten

- Maskeninterpreter
- Treiberbaustein zum Betrieb des Sichtgerätes
- Nutzdaten (auch Nettodaten)-verwaltung auf PSD

} on-line Komponenten

und schließlich

- Sichtgeräteprogramm
  das in der Regel vom Anwender erstellt und auf seinen speziellen Anwendungsfall zugeschnitten wird.

Abhängig von der Aufgabenstellung können neben der Nutzdatenverwaltung noch weitere Komponenten angeschlossen werden, wie später gezeigt wird.

2.1 Maskensprache

Mit der Maskensprache können Bildschirmformate sowohl in ihrem statischen Teil - das ist der darzustellende Textrahmen - wie dem erwarteten dynamischen Teil - das sind die einzugebenden Nutz- oder Nettodaten beschrieben werden.
Der erwartete dynamische Teil, also die Eingabe des Bedieners muß gewissen Anforderungen genügen, die in einer Folge von Prüfvorschriften hinterlegt werden. Es ist möglich, sowohl auf formale als auch auf logische Zulässigkeiten von Eingaben zu prüfen.
Die formale oder Formatprüfung beinhaltet die Untersuchung auf alphanumerische,
alphabetische oder
numerische Eingabe (auch mit Vorzeichen).

Die logische Zulässigkeit - d.h. das Abprüfen von Bedienungsfehlern oder Verträglichkeit der Eingabeparameter untereinander etc. - ist in Form von Vergleichen und Verzweigungen auf anwenderspezifische Subroutinen oder Listeninhalte möglich.

Vergleiche können vorgenommen werden zwischen
- Bildschirmfeldern und Tabellen
- Bildschirmfeldern und Akkumulatoren
- Tabellen und Akkumulatoren
- Akkumulatoren und Akkumulatoren.

Akkumulatoren sind Softwareregister, die sowohl Speicher- als auch Ladeoperationen in Verbindung mit Bildschirmfeldern zulassen.
Darüberhinaus sind Verknüpfungen der Akkus untereinander entsprechend den vier Grundrechnungsarten erlaubt. Die Arithmetik ist dabei für maximal 20 Stellen ausgelegt.

Zur Erstellung von Bildschirmmasken sind Eingabedaten in Lochkartenformat vorgesehen.

Als Eingabekarten für den Maskencompiler werden 6 Kartenarten unterschieden, von denen drei, nämlich die T-, V- und B-Karten mit einigen ihrer Funktionen kurz erklärt werden. (Bild 2 u. 3)

T-Karten dienen zur Formulierung von festen Formularzeilen (Texten) auf dem Bildschirm. Die Texte sind automatisch geschützt und können nicht verändert werden.

V-Karten dienen zur Formulierung von variablen Feldern, d.h. von Nettodatenfeldern.
Innerhalb der Beschreibung eines Nettodatenfeldes können vom Anwender die bereits erwähnten Operationen (Speichern, Laden, Vergleichen) sowie bedingte und unbedingte Sprünge abgesetzt werden.

Durch Verweise auf B-Karten kann erreicht werden, daß das Verlassen eines Nettodatenfeldes erst dann möglich ist, wenn vom Anwender vorgegebene Bedingungen eingehalten wurden.

Ferner ist die Angabe eines Attributes möglich. Mit demAttribut kann die Helligkeit (D = dunkel tasten, H = halbhell) angegeben sowie die Kennzeichnung vorgenommen werden ob es sich um ein Meßfeld, (M) Vollständigkeitsfeld (V) oder geschütztes Feld (G) handelt. Schließlich kann das Format angegeben werden, z.B. N = numerisches Feld ohne Vorzeichen.

B-Karten dienen zur Formulierung von logischen Verknüpfungen zwischen den einzelnen variablen Feldern einer Maske.
Der Verweis auf anwendereigene Routinen und Tabellen ist möglich, womit die Maskenbearbeitung zur Laufzeit beeinflußt werden kann.

2.2 Maskencompiler

Die in der Maskensprache beschriebenen Formulare (Festtext und Nettodaten) werden vom Maskencompiler übersetzt. Dieser Erzeugt als Ausgabestring ein Bitmuster im Urladeformat des PR 310. Die Ausgabe der Masken im Assemblerformat und somit das Einbinden der Masken in das Anwenderprogramm ist ebenfalls möglich.

Als Protokolle (auf Drucker) liefert der Compiler neben einer Liste der Eingabekarten, einem Abbild der erzeugten Masken und einer Fehlerliste noch eine Liste der Nettodatenfelder. (Bild 4)

2.3 Sichtgeräteprogramm, oder Schnittstelle zwischen Maskeninterpreter und Bediener am Sichtgerät

Das Sichtgeräteprogramm erstellt in der Regel der Anwender. Es muß folgende Teilkomponenten enthalten (siehe Bild 2)

- Puffer zur Aufnahme einer Maske (vom Maskencompiler gelieferter Code)
- Puffer für die Eingabedaten (Nettodaten)
- Ein/Ausgabeaufrufe für Sichtgeräte, Drucker, Platte
- variabler Interpreterteil (wird über einen Makro abgesetzt) er enthält: ORG-Aufrufe, Hilfszellen und die vom Anwender benutzbaren Akkumulatoren 15 Stück à 10 Worte
- Aufruf des Maskeninterpreters (als Makro verfügbar)

  Der invariante Interpreterteil ist als Common Code vorhanden, kann also von mehreren Programmen gleichzeitig benutzt werden.

- technologiespezifische Verarbeitung der Nettodaten.

Die Auftrennung von invarianter Maskeninformation (Binäre Maske) und variablen Daten (Netto- oder Nutzdaten) hat den Vorteil, daß der Anwender zur Laufzeit nur mehr die Struktur der Nettodaten kennen muß.

2.4 Interpreter

Beim Eintippen der variablen Daten am Sichtgerät führt der Interpreter Prüfungen auf jedes einzelne Zeichen durch (α-num., numerisch, alphabetisch, Steuerzeichen).

Sind die eingetippten Daten zwar innerhalb des alphanumerischen Zeichenvorrats widersprechen aber dem Feldcode (num., alphabet.), so werden sie am Bildschirm nicht dargestellt.
Als Ersatzzeichen ertönt dafür ein Piepton (Zeichen "BEL").

Eine weitere Eingriffsmöglichkeit in den Datenerfassungsprozeß besteht für den Anwender durch Betätigung von Funktionstasten; diese lösen z.B. die Hardcopy einer am Bildschirm anstehenden Maske aus oder steuern die Cursorbewegung.
Funktionen, die nicht standardmäßig vorgesehen sind, können vom Anwender selbst programmiert werden.

3. Anwendungsbeispiel für Maskensoftware

Zum Abschluß sei noch ein konkreter Anwendungsfall der Maskensoftware aufgezeigt.

Es liege folgende Aufgabenstellung vor:

a) Interaktive Quellprogrammerstellung und Quellprogrammkorrektur am Arbeitsplatz des Programmierers unter Einsatz von simultan arbeitenden Sichtgeräteterminals

b) Formulierung von Arbeitsaufträgen zur Weiterbehandlung von Quellsprache wie Assemblieren, Compilieren direkt am Bildschirmterminal (Remote Job Entry-Betrieb)

Es werden also Betriebsmittel gebraucht, die zum einen einen ständigen Dialog Rechner/Benutzer unterstützen und zum anderen solche Betriebsmittel, die einmal angestoßen, selbsttätig und ohne Kommunikation mit dem Benutzer ablaufen.

Die Überlegungen, mit welchen Hardwarebetriebsmitteln gearbeitet werden soll, könnte zu folgender Rechnerkonfiguration führen (Bild 5)

Hierbei werden alle interaktiven Funktionen und die Auftragsverwaltung, Auftragsprioritierung etc. von einem Konzentratorrechner (hier eine 310) übernommen, während die Batchfunktionen (Weiterbehandlung der eingegebenen Daten) wie z.B. Assemblieren bei einem Zentralrechner (hier einer 330) liegen.

Anschließend daran können die erforderlich werdenden Datenein/ ausgaben und die Frundfunktionen des Systems, die auf dem PR 310 realisiert werden müssen, aufgezeigt werden.

a1) Einbringen der Quelldaten ins System
- Datenerfassung über das Sichtgeräteterminal
- Einbringen von Daten über Floppydisk

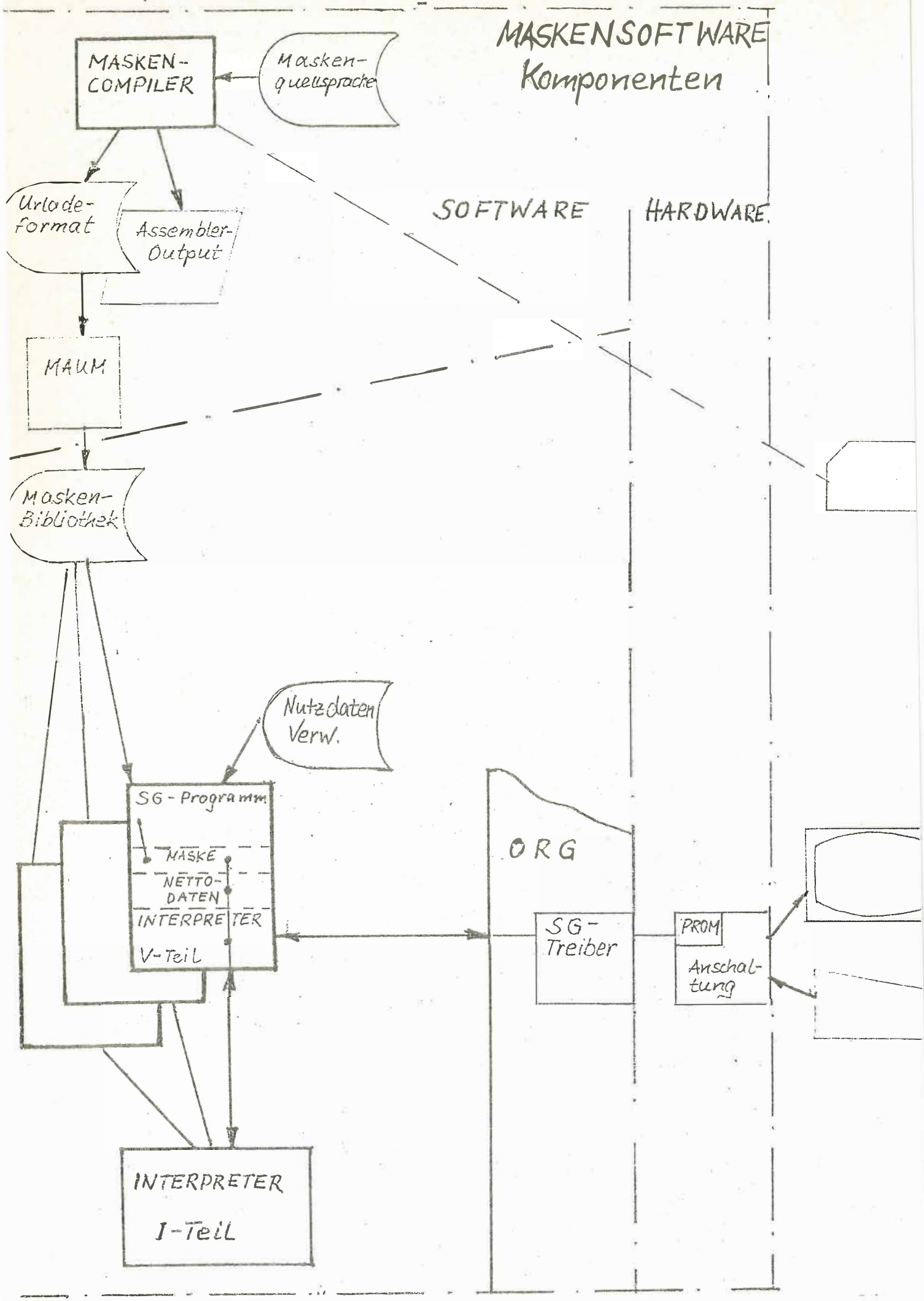
a2) Bedienungseingaben
- Formulierung von Arbeitsaufträgen
- Dateimanipulation

b) Bearbeiten der Daten im System (Grundfunktionen)
- Aufbau und Korrektur der Benutzerdaten
- Verwalten der Benutzerdaten und -Aufträge
- Anstoß der Weiterverarbeitung der Daten mit vorhandenen Dienstprogrammen (Batchprocessing)

c) Ausgabe
- von Daten:
  . auf Floppy, PLS, Drucker
- von Information:
  . z.B. Auftragsstand (wartend, erledigt, storniert, abgebrochen etc.)
  . Dateiübersicht
  . Systembenutzungszeit

Man erkennt also, daß
- die Aufgaben:
  Datenerfassung am Terminal sowie Aufbau u.Korrekturen von Benutzerdaten am Terminal reine Editierfunktionen sind
- die Aufgabe:
  Verwalten der Benutzerdaten u. -Aufträge eine Erweiterung der bisherigen Maskensoftwarekomponenten ist und
- alle übrigen Aufgaben mit Hilfen von Masken formuliert werden können. (Bild 6)

Somit läßt sich die Aufgabenstellung durch Komplettieren des Bildes (2), wie in Bild 7 dargestellt realisieren.


MASKENSOFTWARE
Komponenten
MASKEN-
COMPILER
Masken-
quellsprache
Urlade-
format
Assembler-
Output
SOFTWARE
HARDWARE
MAUM
Masken-
Bibliothek
Nutzdaten
Verw.
SG - Programm
MASKE
NETTO-
DATEN
INTERPRETER
V-Teil
ORG
SG-
Treiber
PROM
Anschal-
tung
INTERPRETER
I-Teil

```
SIEMENS 320/330   DIPOS  CA3     BLATT:      1    DATUM:

GERAET: LKAE(01,00)

     1  /#GMASKE
     2  M TZS
     3  B01    :EQ: T01
     4  B02    :EQ: 'E'/'K'/'G'/'I'
     5  B03    :EQ: T02
     6  B04    :EQ: 1/2/8/9/3/5/6/7
     7  B05    :GE:3 & :LT:8
     8  B06     B:EQ:0
     9  B07      S00
    10  T0101   'SIEMENS SYSTEM 300 T Z S'
    11  T0301   'BENUTZERANMELDUNG - KURZZEICHEN'
    12  T0339     'ANZ:'
    13  T0350     'MES:'
    14  T0401     'NAME:'
    15  T0415     'ABTLG:'
    16  T0429     'TAETIGK:'
    17  T0439     'KONTO:'
    18  T0465     'POS.NR:'
    19  T0701 <I>'AUFTRAEGE'
    20  T0804 <I>'ASSEMBLER                          AS'
    21  T0904 <I>'MAKROGENERATOR                     SM'
    22  T1004 <I>'BINDER                             BD'
    23  T1104 <I>'FORTRAN COMPILER                   FC'
    24  T1204 <I>'TRANSFER                           TR'
    25  T1304 <I>'ANWENDERPROGR.INIT.                RU'
    26  T1404 <I>'DATEI HANDLING                     DH'
    27  T1504 <I>'AUFTRAGSHANDLING                   AH'
    28  T1604 <I>'COMGO                              CG'
    29  T1704 <I>'EDITOR                             ED'
    30  T1804 <I>'PRINT                              PR'
    31  T1904 <I>'MESSAGE                            ME'
    32  T2004 <I>'LOG OFF                            LO'
    33  T2220     'AUFTRAGSWAHL:'
    34  V0333 2<D> """ B01
    35  V0343 6     " C:V-1 X:=C B07*2233 SPR#1 ""
    36  V0354 20    " D:V-1 X:=D SPR#1 ""
    37  V0406 8<M> """
    38  V0421 7<M> """
    39  V0437 1<M> ""N"B04  B05#2 B:=0  SPR*0472  B:=X
    40  V0445 1<M> ""A"B02
    41  V0446 4<MV>""N"
    42  V0451 1<M> ""A"
    43  V0453 1<M> ""A"
    44  V0454 3<MV>""N"
    45  V0458 6<MV>""N"
    46  V0472 5<M> "B06*2233"N"
    47  V2233 2<M> """ B03 A:=X  A:V+
    48  Z
    49  /#  LETZTE KARTE
```

Bild 2

```
MASKE   TZS


                1         2         3         4         5         6         7         8
       12345678901234567890123456789012345678901234567890123456789012345678901234567890


    1  SIEMENS SYSTEM 300 T Z S
    2
    3  BENUTZERANMELDUNG - KURZZEICHEN **     ANZ:****** MES:******************
    4  NAME:******** ABTLG:******* TAETIGK:* KONTO:***** - **** ****** POS.NR:*****
    5
    6
    7  AUFTRAEGE
    8     ASSEMBLER                        AS
    9     MAKROGENERATOR                   SM
   10     BINDER                           BD
   11     FORTRAN COMPILER                 FC
   12     TRANSFER                         TR
   13     ANWENDERPROGR.INIT.              RU
   14     DATEI HANDLING                   DH
   15     AUFTRAGSHANDLING                 AH
   16     COMGO                            CG
   17     EDITOR                           ED
   18     PRINT                            PR
   19     MESSAGE                          ME
   20     LOG OFF                          LO
   21
   22                    AUFTRAGSWAHL:**
   23
/#
/#AMASKE
```

QUL.BIBL: QSB EING.ELEMENT: GMASKE AUSG.ELEMENT: MASKI

LISTE DER NETTODATENFELDER

NR	NAME	ADRESSE	LAENGE
1	V0333	0	2
2	V0343	2	6
3	V0354	8	20
4	V0406	28	8
5	V0421	36	7
6	V0437	43	1
7	V0445	44	1
8	V0446	45	4
9	V0451	49	1
10	V0453	50	1
11	V0454	51	3
12	V0458	54	6
13	V0472	60	5
14	V2233	65	2

MAXIMALE NETTODATENADRESSE: 67

Hardware konfiguration

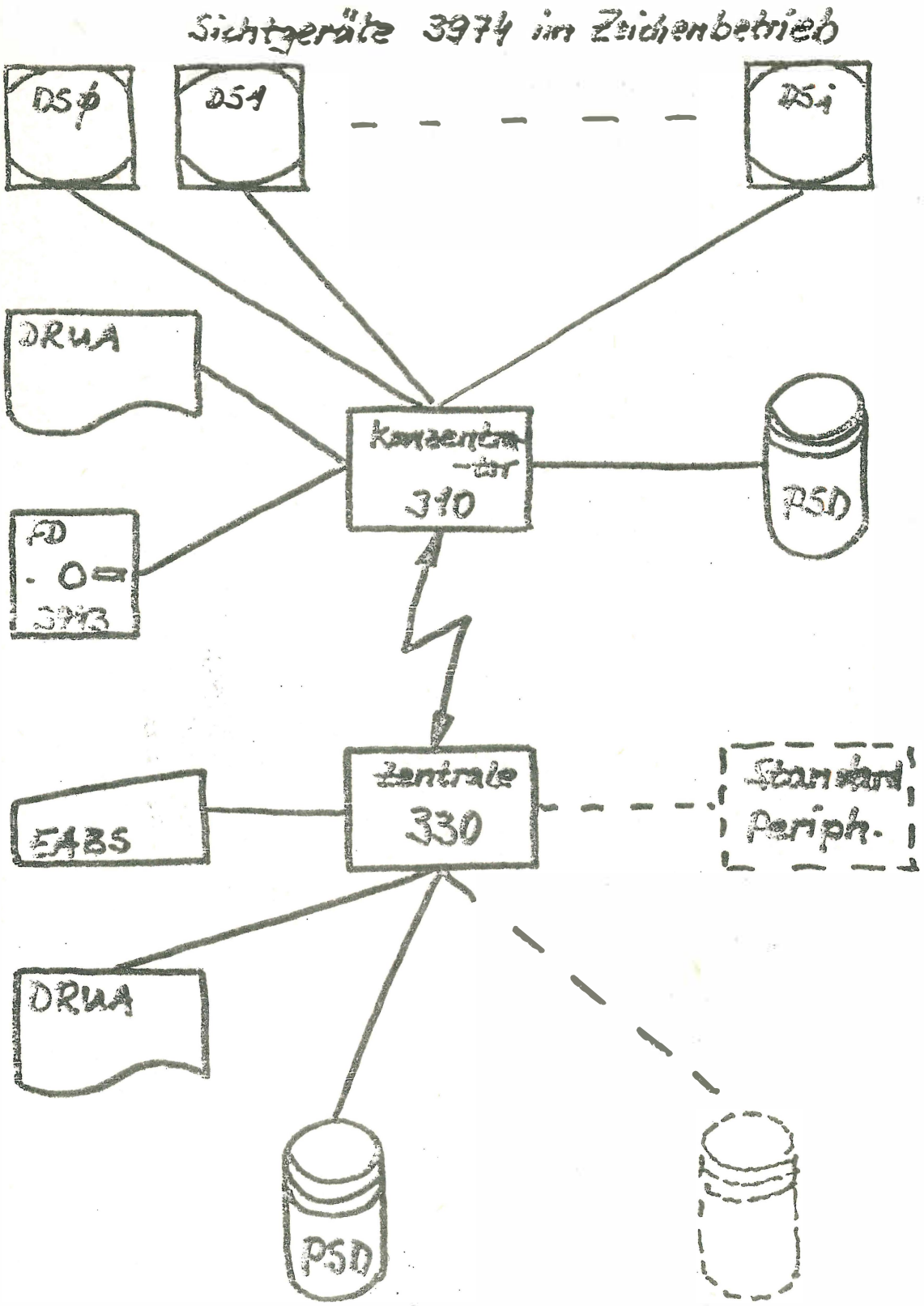


Bild 5

GRUNDMASKE KURZZEICHEN: __

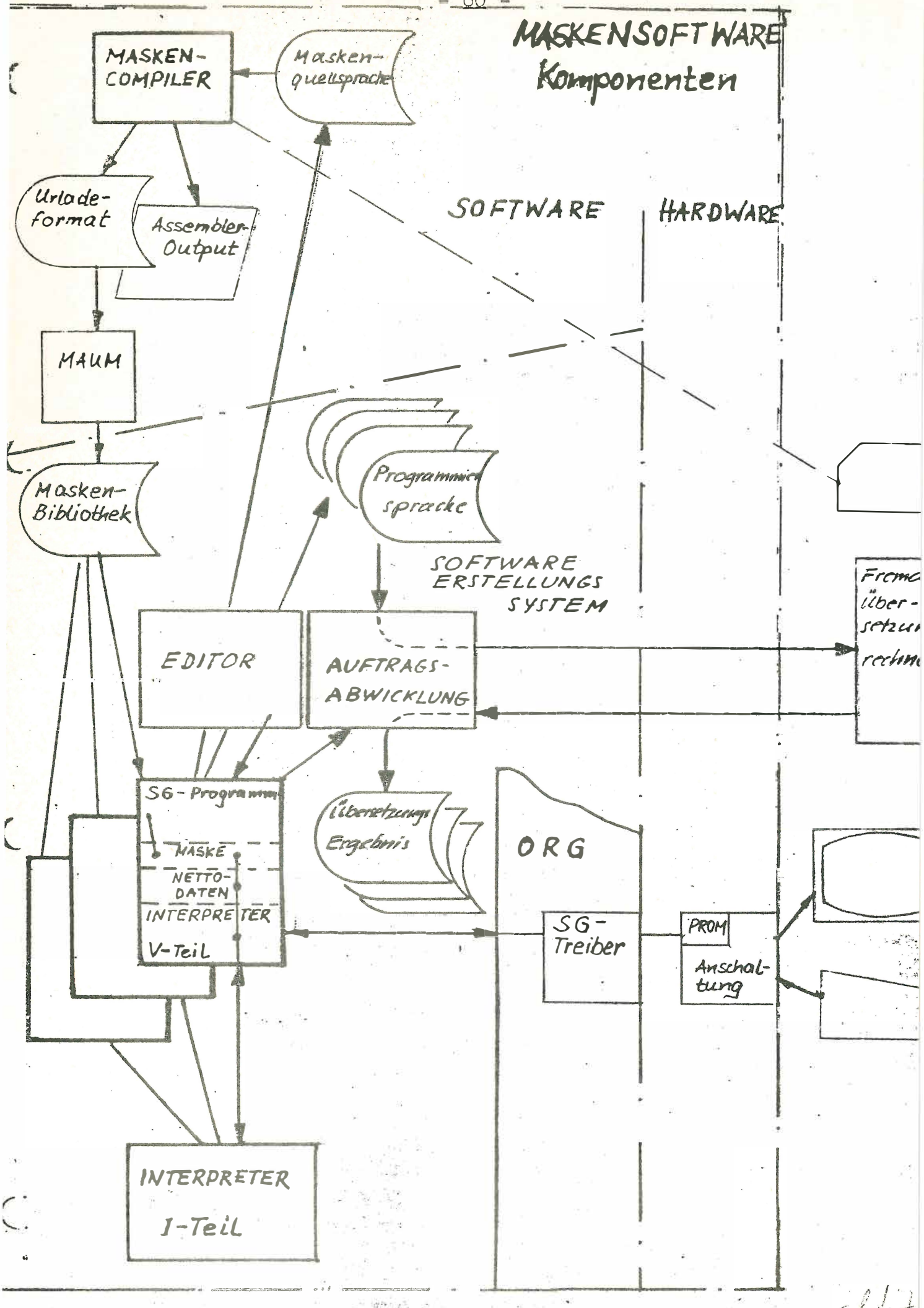
ANZ: _____ MES: _____

NAME: _____ ABTLG: _____ KONTO: _____ _____ _____

AUFTRAEGE

ASSEMBLER	AS
MAKROGENERATOR	SM
BINDER	BD
FORTRAN COMPILER	FC
TRANSFER	TR
ANWENDERPROGR. INIT.	RU
DATEI HANDLING	DH
AUFTRAGSHANDLING	AH
COMGO	CG
EDITOR	ED
PRINT	PR
MESSAGE	ME
LOG OFF	LO

AUFTRAGSWAHL: __


MASKENSOFTWARE
Komponenten
MASKEN-COMPILER
Masken-quellsprache
Urlade-format
Assembler-Output
SOFTWARE
HARDWARE
MAUM
Masken-Bibliothek
Programmier-sprache
SOFTWARE ERSTELLUNGS SYSTEM
EDITOR
AUFTRAGS-ABWICKLUNG
SG-Programm
MASKE
NETTO-DATEN
INTERPRETER V-Teil
Übersetzungs-Ergebnis
ORG
SG-Treiber
PROM
Anschal-tung
INTERPRETER J-Teil

Kurzfassung des Referats:

Neue Kopplungsmöglichkeiten mit dem
Siemens System 300
Witte Siemens AG-Erlangen
Schwerpunktanwendungen von Rechnerverbundsystemen

- Datenverbund: Benutzer einer Anlage greift über eine Koppelstrecke auf Daten zu, die in einer anderen Anlage gespeichert sind.

- Funktionsverbund: Benutzer einer Anlage nutzt Hardware- und Softwarefunktionen einer gekoppelten Anlage z.B. Plotter, Peripherspeicher, Compiler etc.
  auch: Aufteilung von komplexen Aufgaben in sinnvolle Teilaufgaben, die von verschiedenen Anlagen autonom bearbeitet werden Gesamtergebnis wird in der Leitanlage ausgewertet.
- Lastverbund: Aufgaben werden bei Überlastung einer Anlage an eine andere Anlage weitergeleitet und dort bearbeitet, ohne daß der Benutzer davon etwas merkt.
- Sicherheitsverbund: Ausfall einer Anlage wird von anderen Anlagen aufgefangen (Doppelrechner-Systeme, Zwei aus drei Sytstem,usw.)

Notwendige Betriebsmittel

1. Hardware: Kopplungseinheiten, Datenübertragungseinrichtungen

2. Software: Kopplungsbaustein im Organisationsprogramm, Prozedurprogramme, Emulationsprogramme (für Fremdrechneranschluß), Steuerprogramme, Vermittlungsprogramme (packet switching).

Betriebsmittel im System 300

Rechnerkopplungseinheiten 3961, 3962, 3963
Peripheriekopplungseinheiten 3967, 3968

Datenübertragungssteuerung 3965

Peripheriekopplungseinheit 3969

Verschiedene Kopplungsbausteine für variablen Komfort im ORG-Programmpaket: ONKOPP, WIEDKOPP

IMS-KOP zu IBM/370, NEA 300 zu Siemens 4004/7.000

Aktuelle Entwicklungen im System 300

1. Erweiterung der DUST 3965 für Kopplungen im EBCDI-Code ⟶ für flexiblere Verbindungen zu Siemens System 4004/7.000 bzw. IBM 360/370

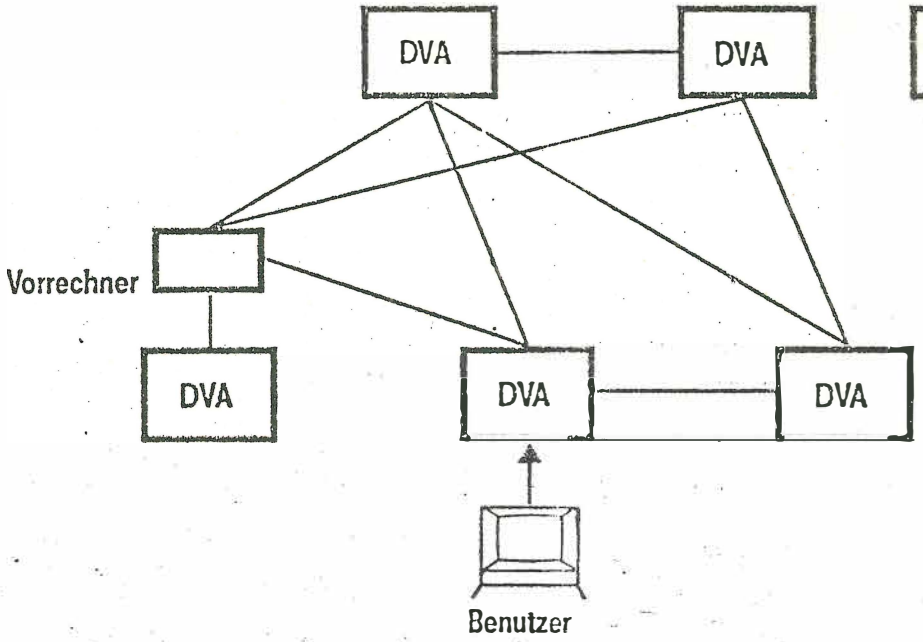
2. Datenübertragungssteuerung 3964
   primär: Kopplung zum Mikrocomputersystem 210
   sekundär: Preisgünstige Punkt-zu-Punkt-Verbindungen zwischen zwei 16 Bit-Rechnern
   ⟶ Möglichkeiten zur universellen Kopplung Fremdsystemen

3. Software: Verschiedene Emulationsprogramme für 4004/7.000-Kopplungen sowie zu IBM 370
   RSPOOL 300, NEA 300 etc.

Weitere Aktivitäten im System 300

- Entwicklung einer HDLC/SDLC-DUST
  mit entsprechenden Leitungsprogramm
  + packet switching

- Einsatz von Lichtleitern zur Datenübertragung
  (zur ultraschnellen Übertragung mit optimaler
  Störunterdrückung ⟶ gegen induktive und kapazi-
  tive Bündelstörungen wie sie speziell im industriel-
  len Bereich häufig auftreten).



Datenverbund

Benutzer einer Anlage greift auf Daten zu, die in anderer Anlage gespeichert sind.

Wichtig für Auskunftssysteme, Platzbuchungssysteme, Dokumentationssysteme.

Funktionsverbund

Benutzer einer Anlage nutzt Hard- und Softwarefunktionen einer anderen Anlage z.B. Plotter, PASCAL-Compiler.

Sehr aktuell im Hochschul- und Großforschungsbereich.

Lastverbund

Wenn eigene Anlage überlastet, wird Auftrag in anderer Anlage bearbeitet. Benutzer merkt davon nichts.

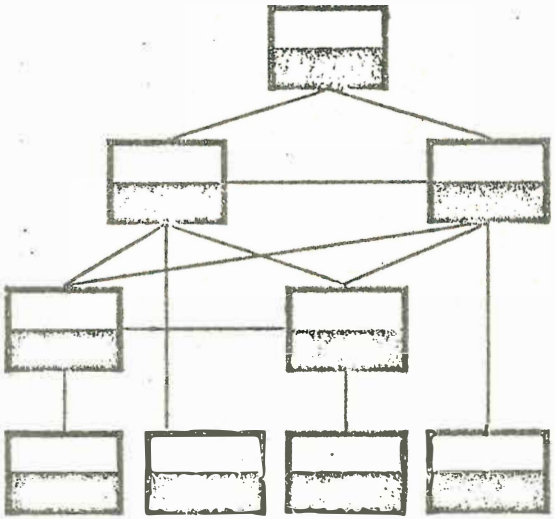
Von Interesse z.B. für DVA-Dienstleistungsunternehmen.

Sicherheitsverbund

Ausfall einer Anlage wird von anderen Anlagen aufgefangen.

Schwerpunktanwendungen eines vermaschten Rechnerverbundsystems

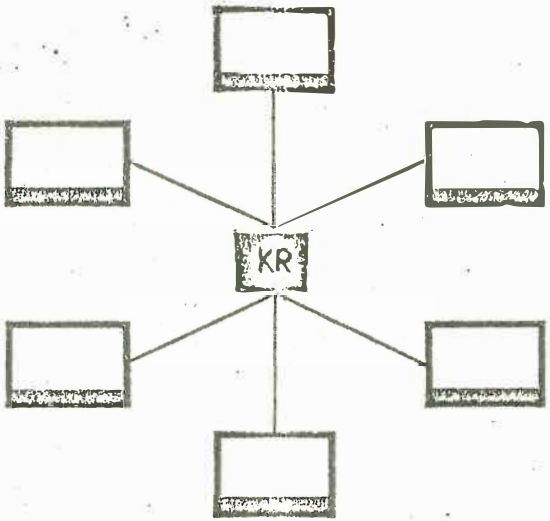
Direkte Verbindung
i.a. mit Teilvermaschung



- schnelle direkte Verbindung
- hohe Belastung der Rechner durch Verbundsteuerung
- aufwendig bzgl. Erweiterung und Test

Einsatz: Prozeßautomatisierung mit hierarchischem Netzaufbau (z.B. GEW Köln)

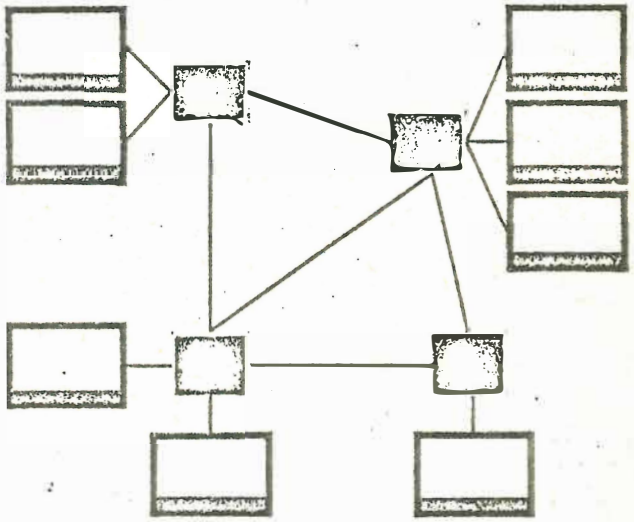
Sternförmige Verbindung mit
zentralem Kommunikationsrechner



- gute Erweiterbarkeit
- Entlastung der Teilnehmerrechner von der Verbundsteuerung
- geringe Ausfallsicherheit der Kommunikation

Einsatz: Großinstitutsbereich (z.B. Technisch Physikalische Bundesanstalt)

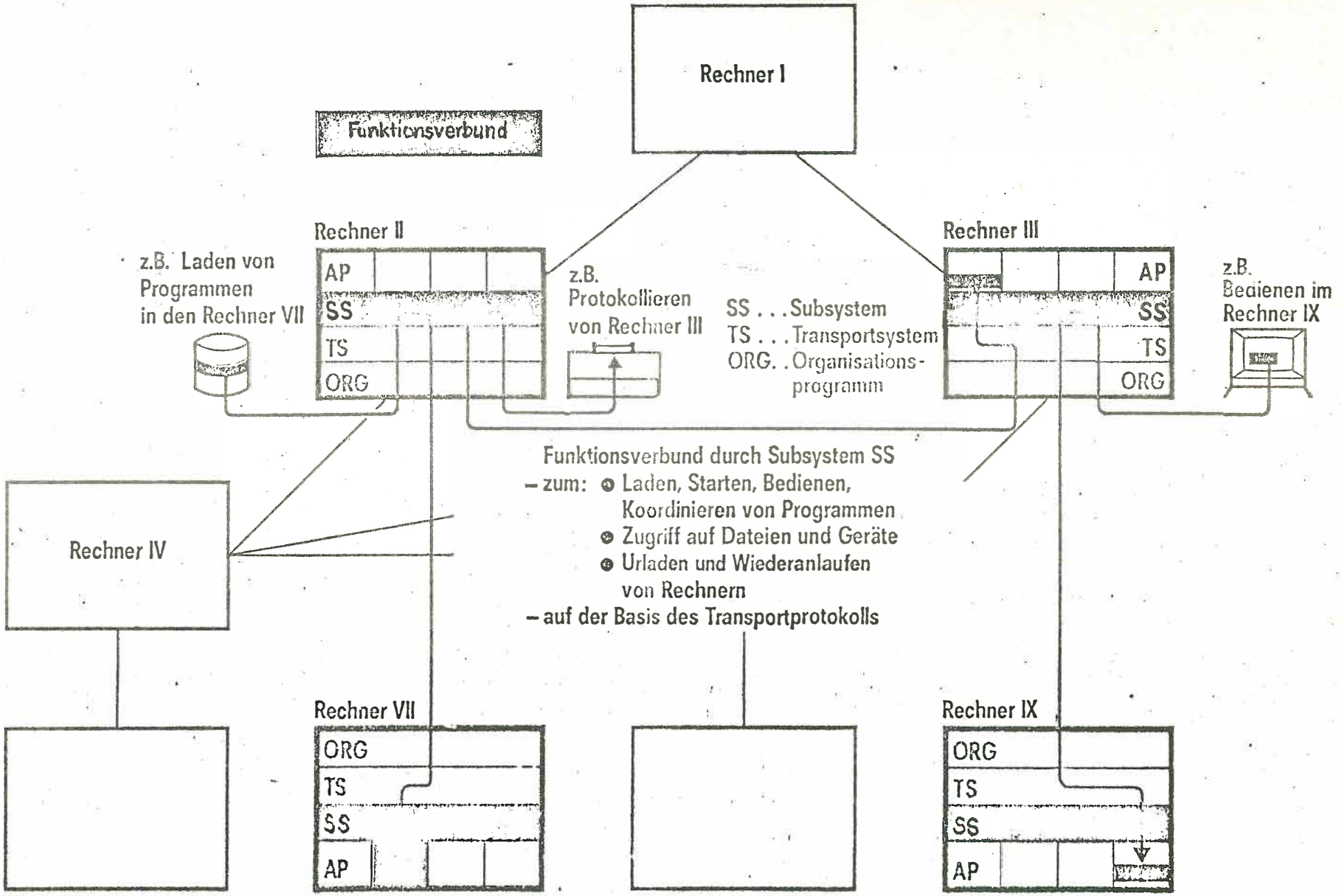
Vermaschte Verbindung mit
verteilten Kommunikationsrechnern



- sehr gute Erweiterbarkeit
- hohe Ausfallsicherheit
- geringe Belastung der Teilnehmerrechner
- niedere Datenraten

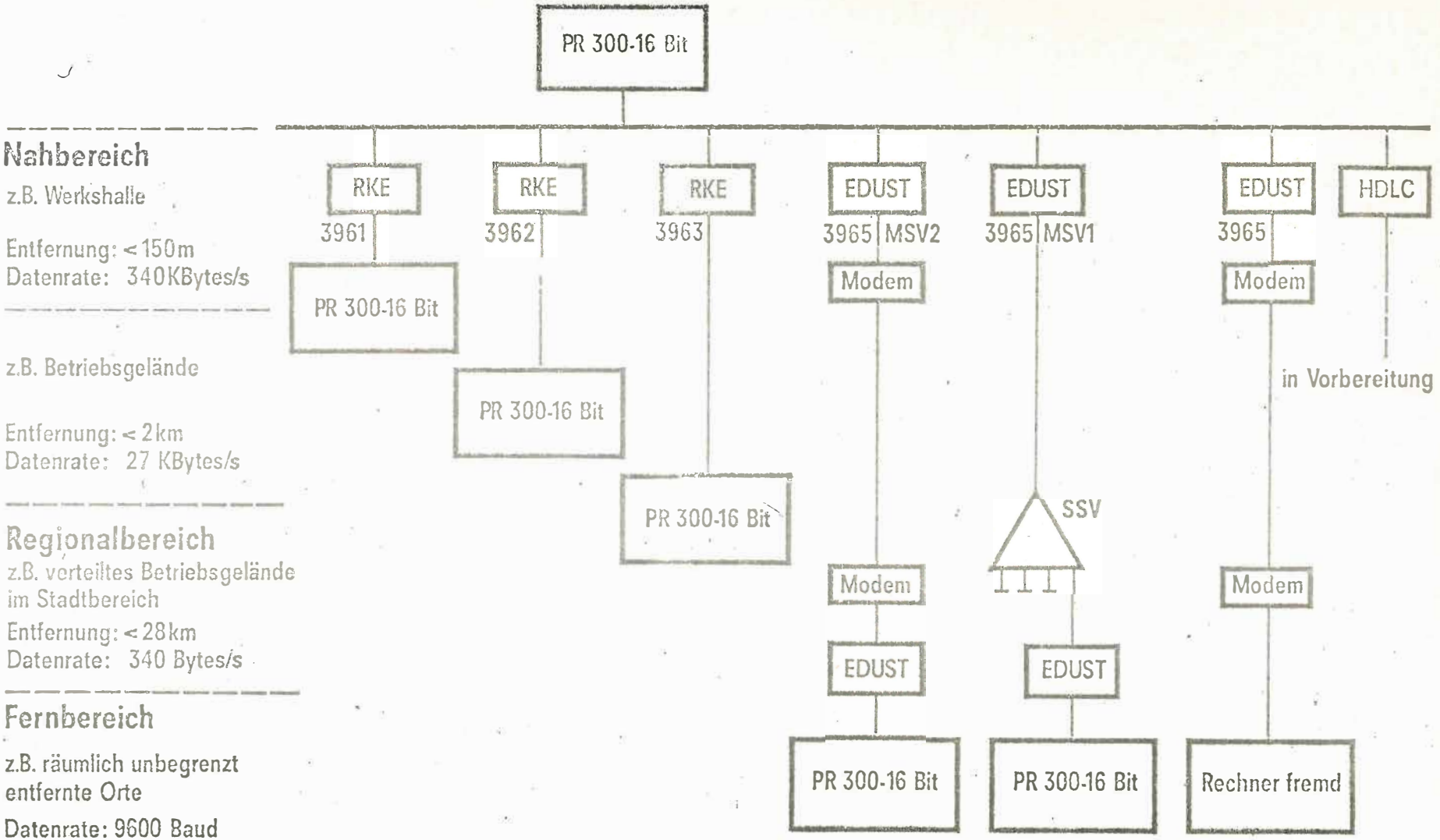
Einsatz: Verbund von Rechenzentren, oder verteilter Prozeßrechneranlagen in der Hüttenindustrie (z.B. HOESCH AG, VOEST)

Gegenüberstellung des Aufbaus von Rechnernetzen

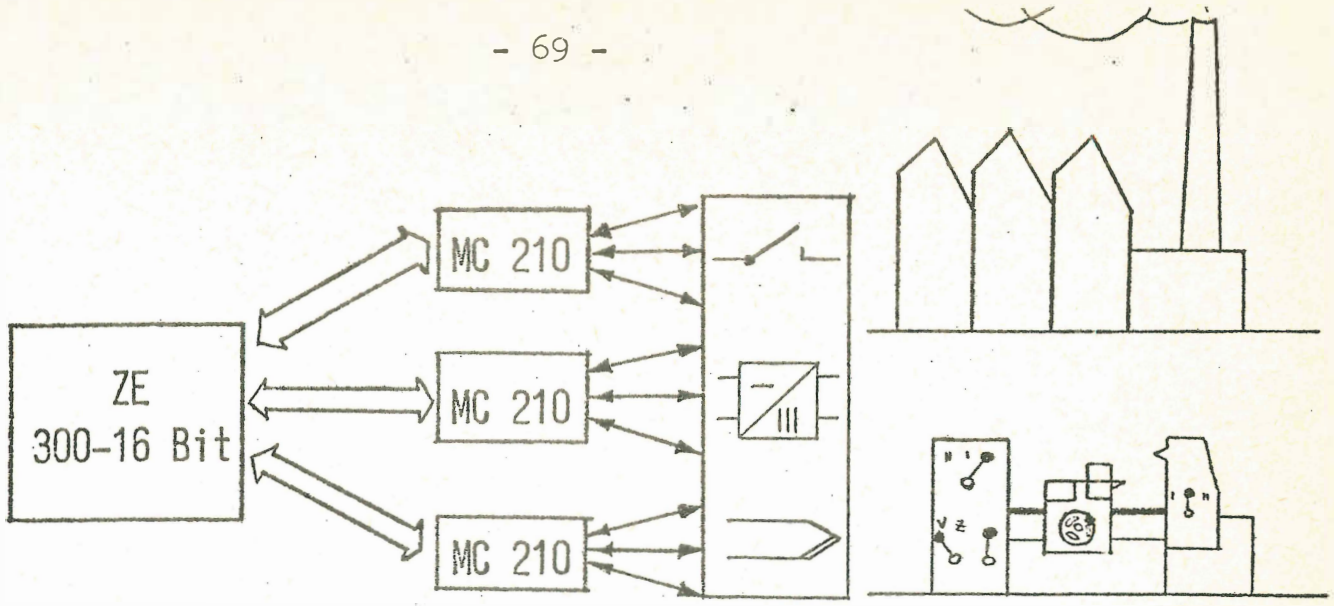

Rechner I
Funktionsverbund
Rechner II
AP
SS
TS
ORG
z.B. Laden von Programmen in den Rechner VII
z.B. Protokollieren von Rechner III
SS . . . Subsystem
TS . . . Transportsystem
ORG. . Organisations-programm
Rechner III
AP
SS
TS
ORG
z.B. Bedienen im Rechner IX
Funktionsverbund durch Subsystem SS
– zum:
Laden, Starten, Bedienen, Koordinieren von Programmen
Zugriff auf Dateien und Geräte
Urladen und Wiederanlaufen von Rechnern
– auf der Basis des Transportprotokolls
Rechner IV
Rechner VII
ORG
TS
SS
AP
Rechner IX
ORG
TS
SS
AP

Datenübertragungseinheiten der Siemens Systeme 300-16 Bit

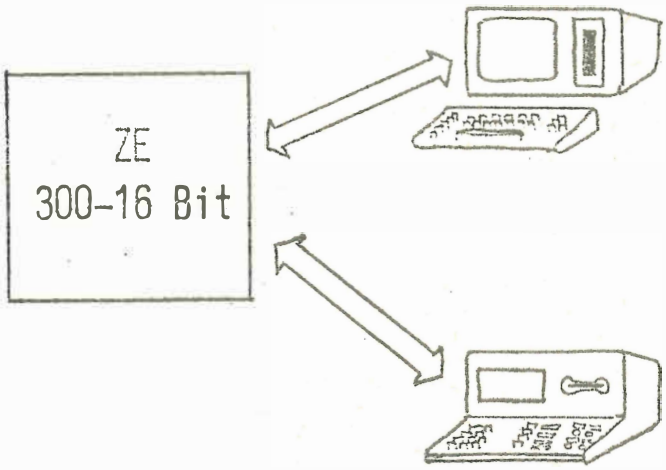
Produkt	Daten	Termine
MC 210-Kopplung DUST 3964	seriell, max. 1 km mit MODEM beliebig max.870 Bytes/s	1/78
Rechnerkopplungs- einheit 3961	parallel, max.150 m max. 350 KBytes/s	lieferbar
Rechnerkopplungs- einheit 3962	seriell, max. 2 km max. 27 KBytes/s	lieferbar
Rechnerkopplungs- einheit 3963	seriell, über GDN max. 28 km max. 340 Bytes/s	lieferbar
Datenübertragungs- steuerung 3965	seriell. über MODEM unbegrenzt max. 9600 Baud	
- Prozedur MSV1	- Mehrpunkt-Verbindung	lieferbar
- Prozedur MSV2	- Punkt-zu-Punkt- Verbindung	lieferbar
Peripheriekopplungs- einheit		
- PKE-E 3967	- seriell, max. 2 km max. 27 KBytes/s	lieferbar
- PKE-B 3968	- seriell, über GDN max. 28 km max. 340 Bytes/s	lieferbar
Peripheriekopplungs- einheit 3969	seriell, Busleitung bis 4 km max. 4 KBytes/s	lieferbar
HDLC-DUST DUST 3966	seriell,beliebige Entfernungen ca. 48 kBaud	8/78



- physikalische und logische Kopplung zwischen allen Rechnern 300-16 Bit
- normierte Datenübertragungsprozeduren für die Kopplung zu systemfremden Rechnern (MSV1/MSV2, in Vorbereitung HDLC)



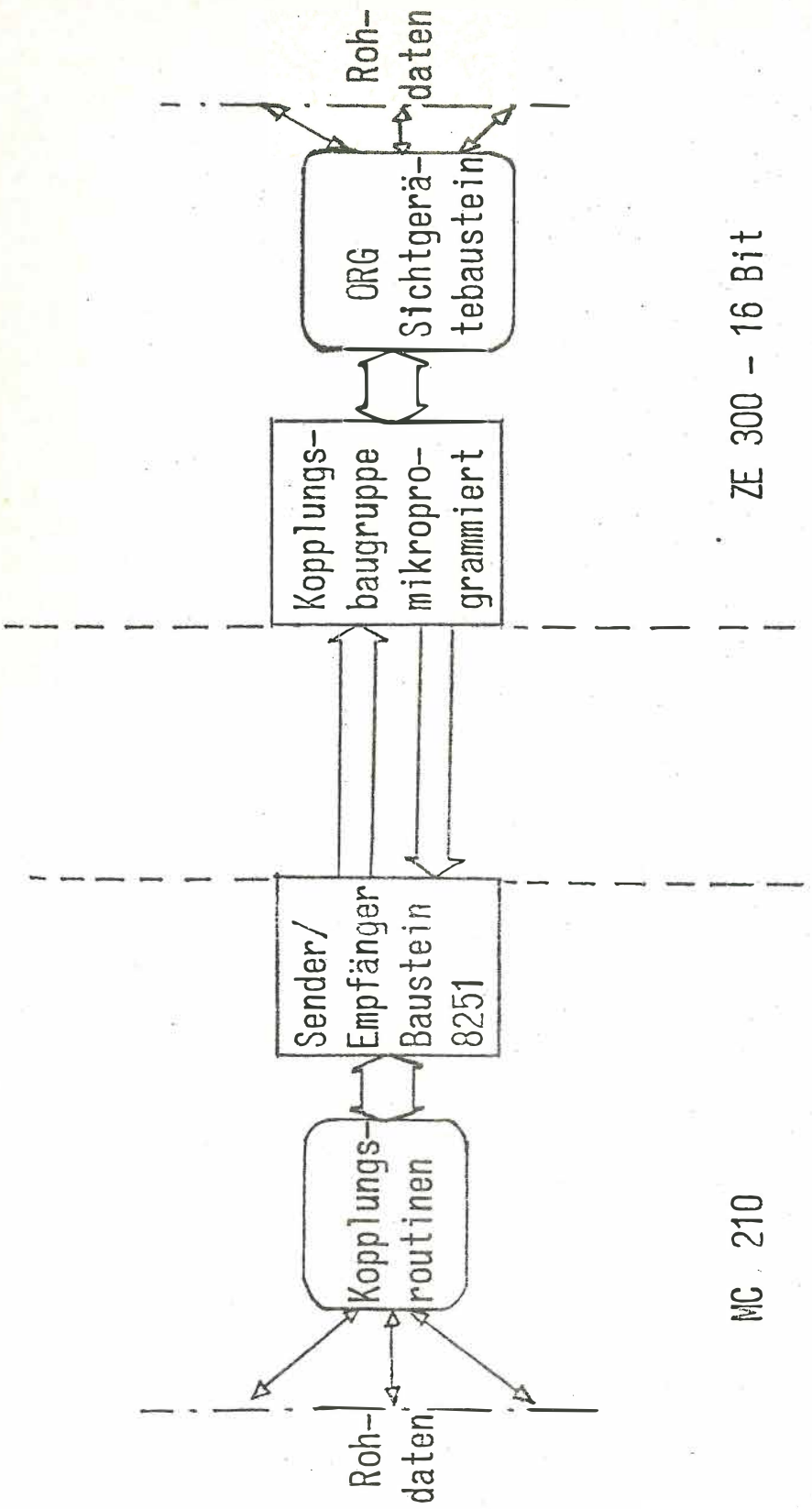
zentraler Prozeßrechner | dezentrale Mikrocomputer | Prozeß-peripherie | Prozeß



Erfassungsstation (Datenerfassungs- und Programmerstellungsplatz mit MC 210)

BDE-Terminal (zur On-line-Erfassung von Fertigungsdaten, Bedienerführung und Vorverarbeitung durch MC 210)

zentraler Prozeßrechner | dezentrale intelligente Terminals


Roh-
daten
ORG
Sichtgerä-
tebaustein
Kopplungs-
baugruppe
mikropro-
grammiert
ZE 300 - 16 Bit
Sender/
Empfänger
Baustein
8251
Kopplungs-
routinen
MC 210
Roh-
daten

LEISTUNGSDATEN DER MC 210-KOPPLUNG

Netto-Datenrate max. 870 Bytes/Sekunde

Auf 4-Draht-Standleitungen für folgende Kopplungen einsetzbar:

- MC 210	mit	MC 210
- MC 210	mit	ZE 300-16 Bit
- (ZE 300-16 Bit	mit	ZE 300-16 Bit)

Potentialtrennung durch Einsatz von

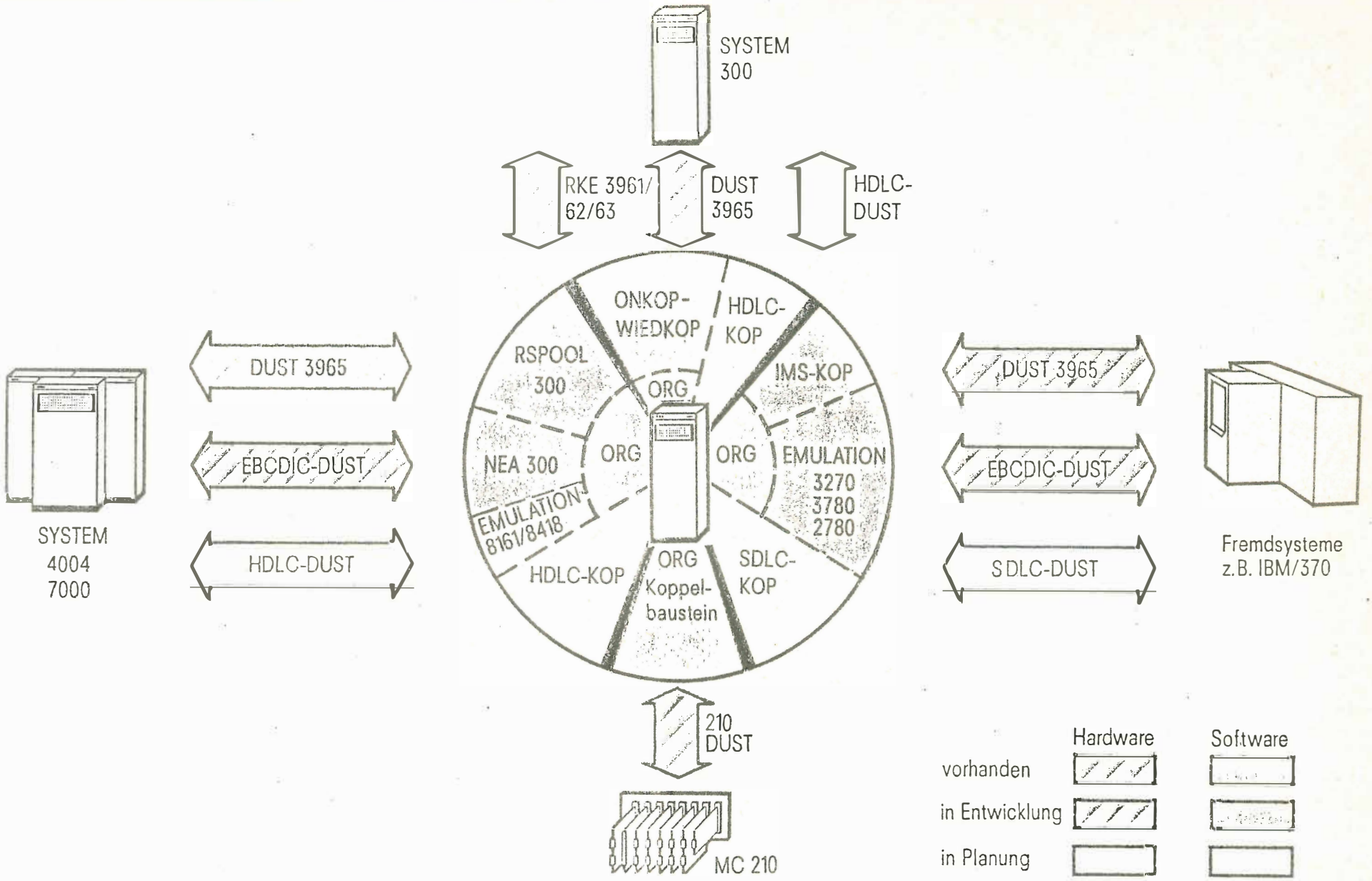
- Optokopplern bei Direktverbindung ≤1 km
- Nahmodem N 10 bei Entfernungen >1 km bis 28 km

Sicherung gegen Fehlerfälle:

- Ausfall des Empfängerrechners
- Unterbrechung der Übertragungsstrecke
- Ausfall von Modems
- Störungen auf der Übertragungsstrecke

durch Quittungsprinzip mit Zeitüberwachungen, Decodierungsschaltungen, Paritätssicherung

Fehlerbehebung durch Wiederholung der gestörten Information



E STE4

Kopplungsmöglichkeiten mit dem SYSTEM 300

Bild 12

 Karlsruhe, den 21.4.1977

Stand der PEARL - Entwicklung

L. Degelow
H.-J. Gottwald
H. Schoknecht

SIEMENS AG Karlsruhe, E STE 333

Inhaltsübersicht

1. Historie der PEARL-Entwicklung

Seit etwa Mitte der sechziger Jahre gibt es Versuche, höhere Programmiersprachen vornehmlich für spezielle Anwendungen in der Echtzeitprogrammierung zu schaffen. Solche in der Praxis eingesetzten Sprache sind zumeist FORTRAN-Derivate. Vielfach behilft man sich auch mit vorgefertigten Programmen, Softwarepaketen, die zum Teil durch sprachähnliche Mittel individuell erweitert wurden (MADAM, ARSS etc.).

Ende der sechziger Jahre tauchten dann vor allem in Europa Sprachentwürfe mit universellem Charakter auf, z.B.:

RTL/2	in England
PROCOL	in Frankreich
PAS1 und PEARL	in der BRD

Im Herbst 1969 trat erstmals der PEARL-Arbeitskreis, eine Arbeitsgruppe aus Anwender, Herstellerfirmen, Softwarehäusern und Instituten, zusammen.

Ziel der Arbeiten war die Definition einer höheren Programmiersprache für leichtere und funktionsgerechtere Programmierung von Echtzeitproblemen. Eine solche Sprache soll auch den Austausch von Prozeß- und Experiment-Programmen erleichtern.

Ein erster Rohentwurf zur Sprache PEARL wurde 1973 vorgelegt. Um eine einheitliche Sprachdefinition bemühten sich von Anfang an die Implementatoren. Diese Einigung wurde im Januar dieses Jahres erzielt und durch die Definition von Basis-PEARL festgeschrieben.

2. Charakterisierung der Echtzeitsprache PEARL

PEARL wird meines Erachtens zu recht als universelle höhere Programmiersprache bezeichnet.

PEARL ist universell im doppelten Sinn:

Sie ist zum einen für ein breites Anwendungsspektrum, zum anderen auch praktisch für alle kommerziell eingesetzten Prozeßrechner geeignet.

Der Trend zu höheren, d.h. maschinenunabhängigen Programmiersprachen ist allgemein akzeptiert. Sie versprechen insbesondere verminderte Programmherstellungskosten.

Die Gründe dafür, daß man sich in der Prozeßdatenverarbeitung so lange mit maschinenabhängigen Sprachen behelfen mußte, sind in der stets vorhandenen Verschiedenartigkeit der Prozeßperipherie und in einigen Besonderheiten der Prozeßrechnersysteme zu suchen, die programmtechnisch schwierig zu beherrschen sind.

Diese Besonderheiten sind - kurz gesagt - folgende:

- Die Programme müssen über spezielle Geräte (Prozeßperipherie) mit dem technischen Prozeß verkehren, die für jedes System individuelle Adressen, Namen und Strukturen haben.

- Viele Programme müssen sich selbst schritthaltend mit der Real-Zeit und ggfs. den Ereignissen in anderen Programmen im Ablauf steuern können (Synchronisationsproblem).

PEARL enthält im Sprachkern Formulierungsmittel für diese Aufgaben.

Im folgenden möchte ich auf die wichtigsten Sprachmittel und die Struktur von PEARL-Modulen eingehen.

3. Aufbau eines PEARL-Moduls

- siehe Bild 1 -

In einem PEARL-Programm sind die Beschreibung der System-Konfiguration (Anschluß der Peripherie = Systemteil) und die Problemformulierung (Problemteil) getrennt.

Diese Trennung hat den Vorteil, daß der Problem-Teil unabhängig von der Anlagen-Konfiguration formuliert werden kann. Durch Hinzufügen (Binden) eines entsprechenden System-Teiles wird das Programm auf der Zielmaschine ablauffähig.

Ein PEARL-Modul wird von den Schlüsselworten MODULE und MODEND eingefaßt. Der Systemteil wird durch das Schlüsselwort SYSTEM eingeleitet. Das Schlüsselwort PROBLEM ist das Ende des Systemteils bzw. leitet den Anfang des Problemteils ein.

Das Schlüsselwort MODEND kennzeichnet das Ende des Systemteils, wenn der Problemteil fehlt.
Es ist möglich den Systemteil oder einen Problemteil getrennt zu übersetzen.

Wird aus den einzelnen Tasks zu globalen Daten zugegriffen, so wird zusätzlich ein Common-Data-Modul mit dem Namen PEARLC generiert.

Ein Problemteil kann aus mehreren Tasks und/oder globalen Proceduren bestehen.

Aufbau eines PEARL-Moduls

```
/#PEARL
     MODULE;
     SYSTEM;                    /* Systemteil */
          .
          .
          .
        Anweisungen;
          .
          .
          .
     PROBLEM;                   /* Problemteil */
     DCL ..... ,
         ..... ;                /* globale Daten */

     T1: TASK;                  /* 1. Task       */
     DCL ..... ;
          .
          .
          .
        Anweisungen;
          .
          .
          .
     END;
      .
      .
      .
     Tn: TASK;                  /* n-te Task     */
      DCL ..... ;
          .
          .
          .
        Anweisungen
          .
          .
          .
     END;
     MODEND;
/=
/*
```

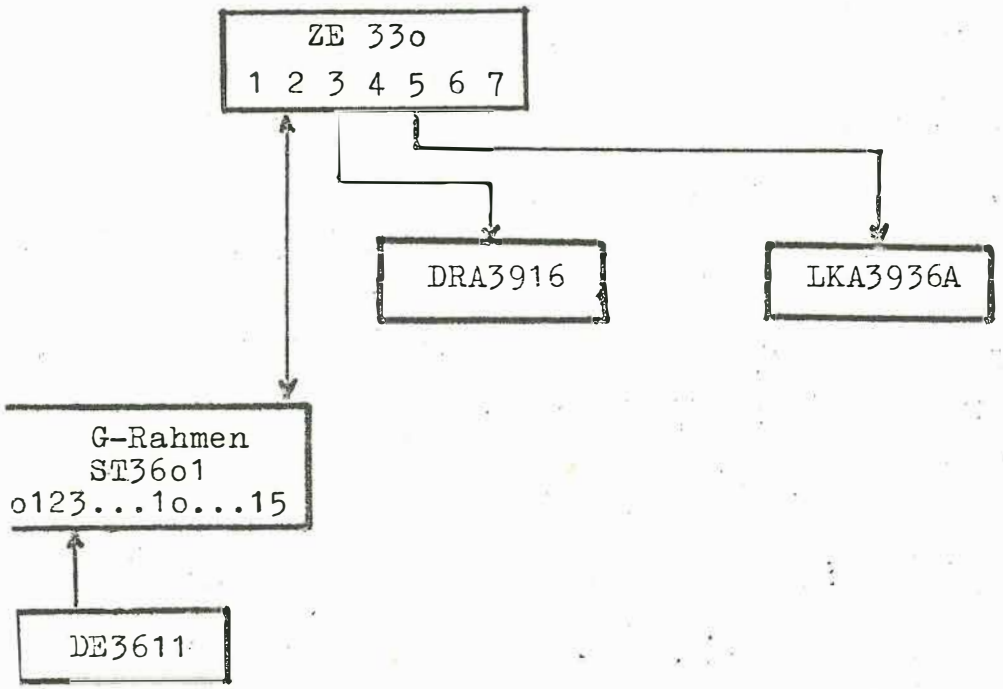
3.1 Der SYSTEM-Teil

Im SYSTEM-Teil wird die gesamte Hardware-Konfiguration der für die PEARL-Programme benötigten Peripherie-Geräte beschrieben.

Den einzelnen Anschlußstellen der E/A-Geräte, Prozeßendstellen, Interrupts und Signals bzw. den Knotenpunkten kann ein Anwendername zugeordnet werden. Dieser Anwendername muß im PROBLEM-Teil spezifiziert werden, damit er in diesem verwendet werden kann. Der SYSTEM-Teil beschreibt die Richtung des gewünschten Datenflusses zwischen den einzelnen Anschlußstellen.

Beispiel für einen SYSTEM-Teil

```
SYSTEM;
ST3601 <--> ZE330 * 2;
DE1:       DE3611 --> ST3601 * 3;
DRUA:      DRA3916 <-- ZE330 * 3;
LKAU:      LKA3936A <-- ZE330 * 5;
```



n die Zentraleinheit ZE330 soll ein Schnelldrucker, eine
ochkartenausgabe direkt und eine Digitaleingabe über einen
-Rahmen (Grundsteuerung) angeschlossen werden.

3.2 Der PROBLEM-Teil

Alle innerhalb des PROBLEM-Teils verwendeten Größen müssen in diesem deklariert bzw. spezifiziert werden. Der gesamte PROBLEM-Teil besteht demgemäß aus dem Modul-Spezifikationssatz, dem Modul-Deklarationssatz, den globalen Proceduren und den **Tasks.**

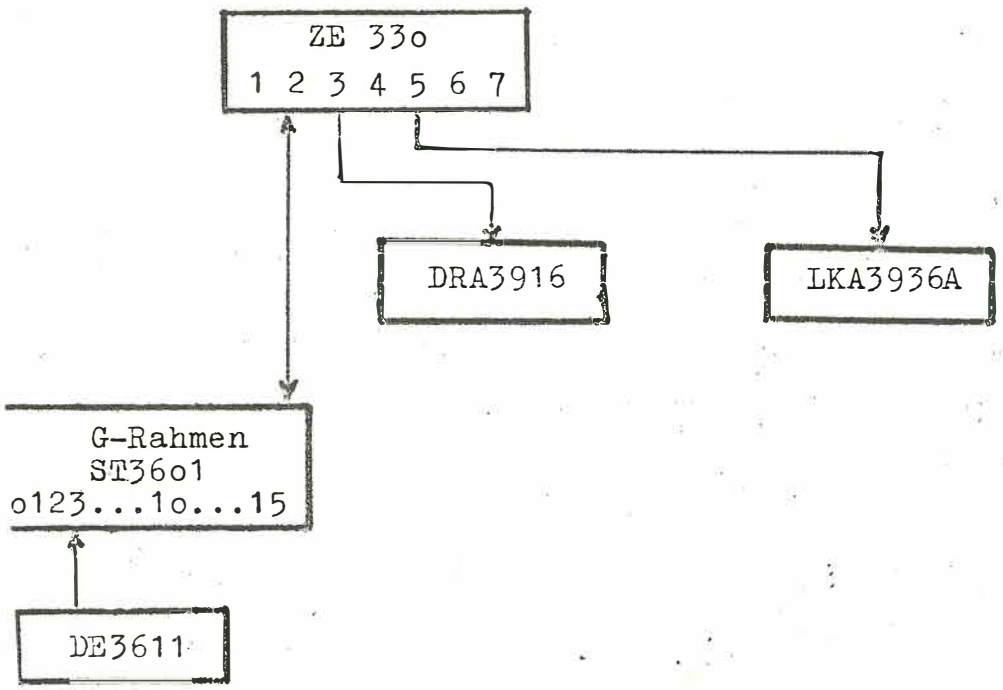
Dabei sind die Tasks Programme im Sinne des Betriebssystems.

Die gesamten arithmetischen Anweisungen, sowie die Anweisungen zur Organisation der Real-Zeit-Verarbeitung können nur innerhalb der Tasks oder Proceduren formuliert werden. Der Problemteil als Ganzes ist nicht ausführbar. Datenaustausch zwischen TASK's erfolgt über globale, mindestens auf Modulebene deklarierte, Daten und Daten-Bereiche. Hierfür stehen dem Anwender alle üblichen arithmetischen Funktionen einer höheren Programmiersprache (FORTRAN, ALGOL, PL/1) zur Verfügung.

Zusätzlich kann er den Datentyp STRUCT verwenden. Dieser Datentyp faßt einzelne Standard-Datentypen zu einem Verbund zusammen. Der Zugriff zu einem einzelnen Element der Struktur ist über die Selektion möglich.

Beispiel für einen SYSTEM-Teil

```
SYSTEM;
ST3601 <---> ZE330 * 2;
DE1:         DE3611 ---> ST3601 * 3;
DRUA:        DRA3916 <--- ZE330 * 3;
LKAU:        LKA3936A <-- ZE330 * 5;
```



n die Zentraleinheit ZE330 soll ein Schnelldrucker, eine
ochkartenausgabe direkt und eine Digitaleingabe über einen
-Rahmen (Grundsteuerung) angeschlossen werden.

3.2 Der PROBLEM-Teil

Alle innerhalb des PROBLEM-Teils verwendeten Größen müssen in diesem deklariert bzw. spezifiziert werden. Der gesamte PROBLEM-Teil besteht demgemäß aus dem Modul-Spezifikationssatz, dem Modul-Deklarationssatz, den globalen Proceduren und den **Tasks.**

Dabei sind die Tasks Programme im Sinne des Betriebssystems.

Die gesamten arithmetischen Anweisungen, sowie die Anweisungen zur Organisation der Real-Zeit-Verarbeitung können nur innerhalb der Tasks oder Proceduren formuliert werden. Der Problemteil als Ganzes ist nicht ausführbar. Datenaustausch zwischen TASK's erfolgt über globale, mindestens auf Modulebene deklarierte, Daten und Daten-Bereiche. Hierfür stehen dem Anwender alle üblichen arithmetischen Funktionen einer höheren Programmiersprache (FORTRAN, ALGOL, PL/1) zur Verfügung.

Zusätzlich kann er den Datentyp STRUCT verwenden. Dieser Datentyp faßt einzelne Standard-Datentypen zu einem Verbund zusammen. Der Zugriff zu einem einzelnen Element der Struktur ist über die Selektion möglich.

Beispiel

Deklaration

```
DECLARE  CM  STRUCT (/ L FIXED, M FLOAT /);
```

Anwendung (Selektion)

```
CM.L : =  15 ;
```

Aufbau eines PROBLEM-Teils:

```
PROBLEM;
SPC K   SIGNAL      /x MODUL-SPEZIFIKATIONSSATZ*/
SPC L   ENTRY (INV FIXED);
     .
     .
     .
DCL X   (3,5) FIXED, /*MODUL-DEKLARATIONSSATZ*/
DCL Y   STRUCT (/S1 BIT, S2 FLOAT/);
     .
     .
     .
   T1 : TASK;
     .
     .
     .
        X (1,4) := 12;
     .
     .
     .
        END;
   P1 : PROC (Z FIXED) RETURNS (FLOAT);
     .
     .
     .
        RETURN (Y,S2);
     .
     .
     .
        END;
     .
     .
MODEND;
```

PEARL-Sprachmittel zur Real-Zeit-Verarbeitung

Da Prozeßsteuerungs-Systeme in der Regel aus einer Anzahl von in sich autonomen Programmen bestehen, die Einzelprozesse steuern, ist eine Koordinierung auf höherer Ebene erforderlich. Um diese asynchron verlaufenden Prozesse zu koordinieren, besitzt PEARL eine Reihe von Sprachelementen.

Arbeiten mit TASK's

In PEARL werden alle Tasks mit ihren bei der Deklaration angegebenen Namen angesprochen.

Dem Anwender stehen Aufrufe zum Starten (ACTIVATE), Beenden (TERMINATE), Anhalten (SUSPEND), Fortsetzen (CONTINUE) einer Task zur Verfügung. Weiterhin kann eine Task angehalten werden; sie wird auf ein bestimmtes Ereignis hin fortgesetzt. Die Aufrufe können mit einer entsprechenden Zeit-Modifikation (Schedule) versehen werden.

Beispiele:

```
AT 5:Ø:Ø EVERY 2HRS ACTIVATE T;
```

Die Task T soll um 5 Uhr ab, alle 2 Stunden zyklisch gestartet werden.

```
WHEN IT1 RESUME;
```

Die laufende Task wird angehalten und bei Eintreffen des Interrupts IT1 fortgesetzt.

Alle mit einem Schedule versehenen Aufrufe einer Task (z.B. mit Namen T1) können vom Anwender zurückgenommen werden.

Beispiel:

```
PREVENT T1;
```

Alarm-, Interrupt- und Signal-Anwendung

Mit Hilfe des oben erwähnten WHEN-Statements kann der Anwender auf das Eintreffen eines Alarms warten.

Weiterhin stehen ihm eine Anzahl Standard-Signals zur Verfügung, die compiler- und systemabhängig sind. Der Anwender hat die Möglichkeit, auf ein Signal durch das ON-Statement zu reagieren. Wünscht er dies nicht, wird eine Standard-Fehler-Routine durchlaufen.

Beispiel:

```
ON SIG1 :   :   /*SIGNAL-REAKTION*/
```

Die Reaktion auf ein Signal kann der Anwender auch durch die folgende Anweisung einleiten:

```
INDUCE SIG1;
```

Bei Eintreffen eines Signals wird die ON-Reaktion durchlaufen und das Programm gegebenenfalls an der Aufrufstelle fortgesetzt.

Semaphore

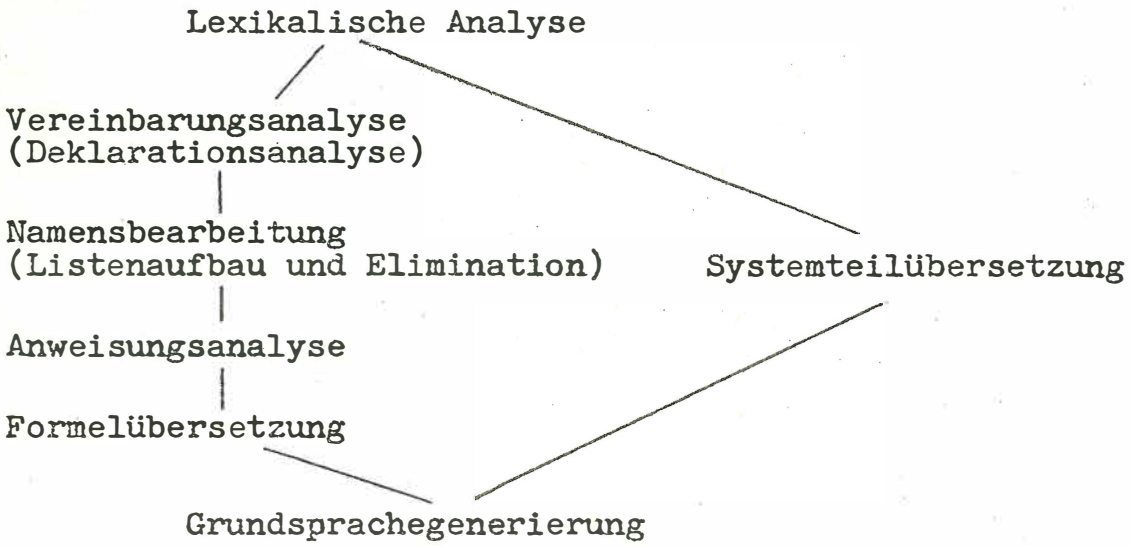
Semaphore stehen dem Anwender zur Koordinierung von Tasks zur Verfügung. Besonders zur Koordinierung der Ein-/Ausgabe-Vorgänge eignet sich die Verwendung von Semaphoren. Zur Koordinierung dienen die REQUEST-Anweisung (Semaphor-Variable erniedrigen) und die RELEASE-Anweisung (Semaphor-Variable erhöhen).

Hat die Semaphor-Variable den Wert Ø und wird eine REQUEST-Anweisung ausgeführt, so wird die aufrufende Task angehalten und fortgesetzt, wenn von einer anderen Task eine RELEASE-Anweisung auf diese Semaphor-Variable durchlaufen wird.

Ist der Wert der Semaphor-Variablen bei Ausführung der REQUEST-Anweisung größer Ø, so wird die Semaphor-Variable um 1 erniedrigt und die aufrufende Task fortgesetzt.

4. Der PEARL-Compiler PC30

Ein in der Sprache PEARL geschriebenes Programm wird vom Compiler in mehreren Durchläufen (= Pässen) auf eine Folge von Befehlen auf Grundsprache-Ebene zurückgeführt. Die einzelnen Pässe haben folgende Aufgaben:



Alle Teile des Compilers werden mit dem Binder BD30 zu einem ablauffähigen System gebunden. Die Arbeitsfassung des PC30 wird mit dem Segmentiersystem SEGSYS erstellt.

Einige Daten des Compilers:

minimaler Laufbereich : 16K Worte

siehe Bild 3

5. Erstellungsweg eines PEARL-Programms

Der PEARL-Compiler liest die Quellsprache wahlweise von Lochkarte bzw. PSD (Bedienparameter) und übersetzt i.a. einen PEARL-Modul in mehrere Grundsprachemodule der Systeme 300-16 Bit.

Die Quellsprache sowie die Grundsprachemodule entsprechen der Norm des Bibliothekssystems BISY.

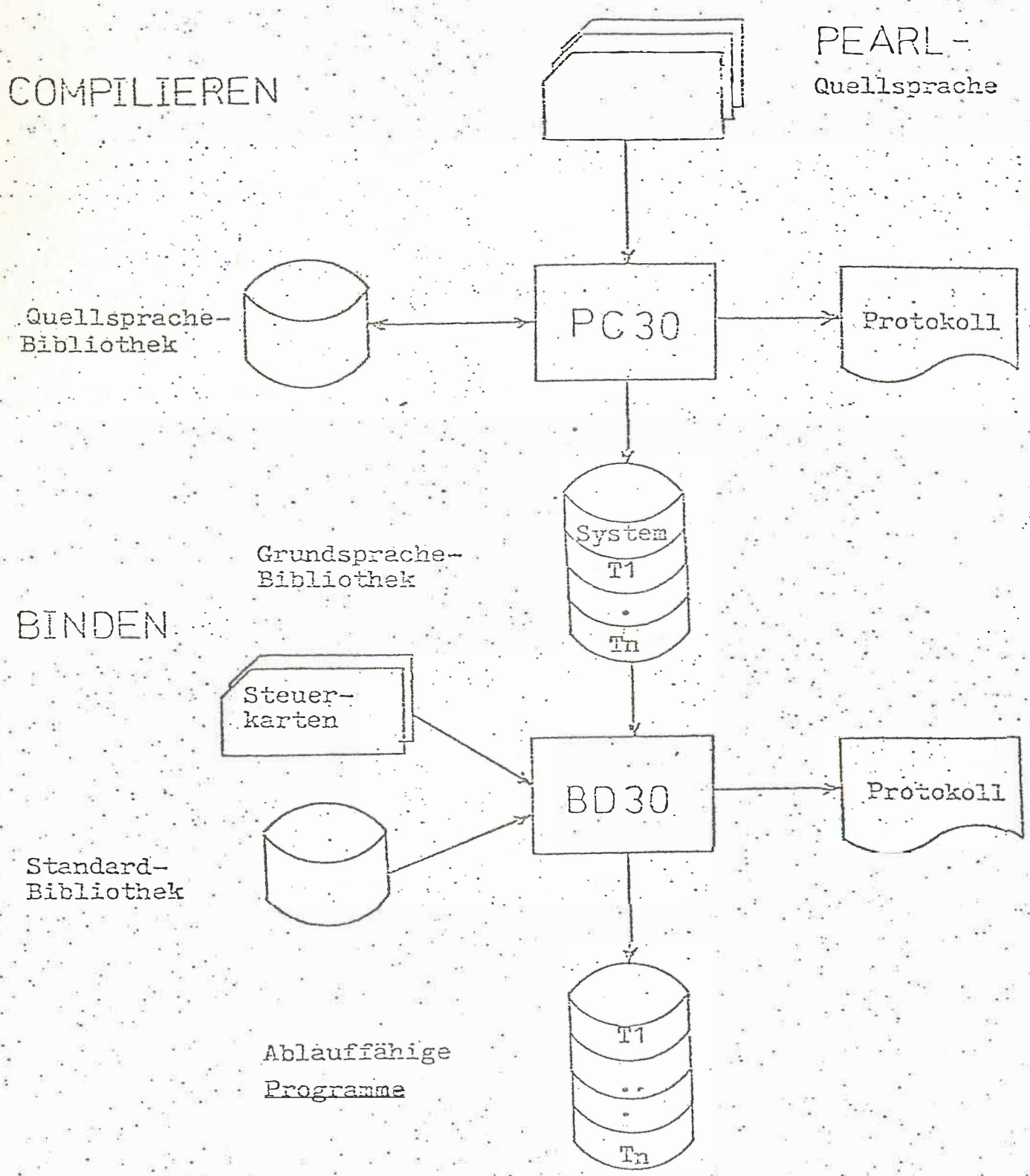
Durch Binden mittels des Binders BD30 der einzelnen Tasks und den entsprechenden Funktionen aus der Standardlibrary entstehen ladbare Grundsprachemodule.

Durch Laden dieser Module mittels des Ladebinders werden die Beziehungen zu den globalen Größen (Variablen) hergestellt und es entsteht ein ablauffähiges PEARL-System.

Aufgrund der Tatsache, daß einzelne Teile eines PEARL-Systems getrennt übersetzt werden können, sind die Steuerkarten für den BD30 vom Anwender zu schreiben. Der PEARL-Compiler teilt beim Übersetzungslauf jedoch die Parameter am Bediengerät mit.

ABLAUFBLOCKBILD Zur

Erstellung eines ablauffähigen PEARL-Programms

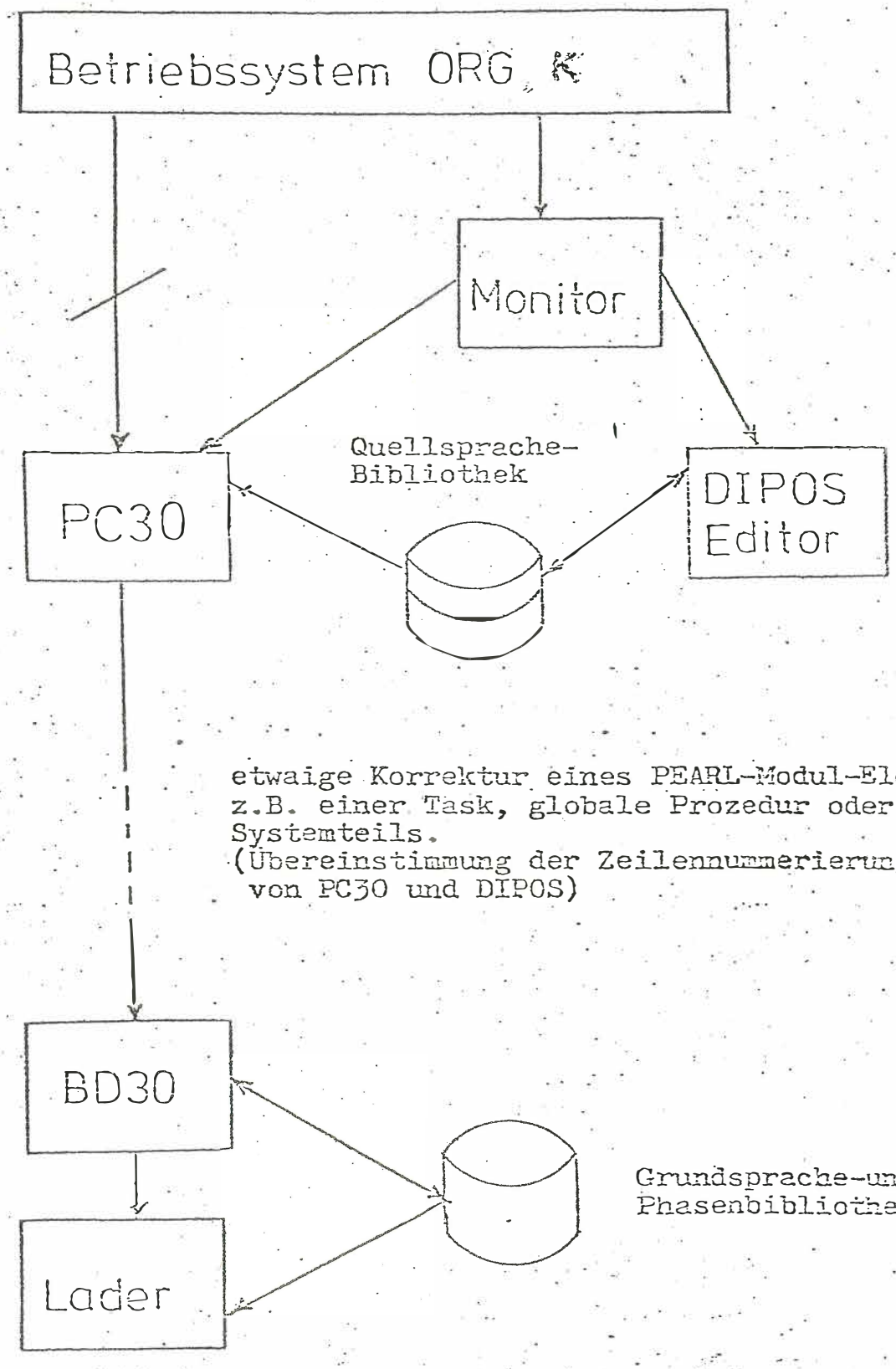


6. Integration des PEARL-Compilers PC30 in das Spektrum der Dienstprogramme SIEMENS 300-16 Bit

Der PEARL-Compiler ist unter dem Betriebssystem ORG 330 K und/oder dem Monitor MONI 330 ablauffähig. Der In- sowie Output lassen sich mit der Standardsoftware bearbeiten. Insbesondere lassen sich die PEARL-Quelldaten mittels Editor (Funktion des DIPOS) korrigieren (Übereinstimmung der Zeilennummerierung).

Integration des PEARL-Compilers PC30

in das Spektrum der Dienstprogramme SIEMENS 300-16 Bit



7. Erwartungen durch Anwendung von PEARL

Aufgrund der immer höher steigenden Erstellungskosten der Software bei gleichzeitigem Absinken der Hardwarekosten, werden mit Verwendung von PEARL die Programmerstellungskosten sinken.

PEARL kann bei

- der Aufgabenanalyse? wenig!
- Systementwurf? einiges!
- Programmtest? einiges!
- Dokumentation? viel!
- Inbetriebnahme? wenig!
- Programm-Wartung? einiges!

zur Verminderung der Kosten und Vereinheitlichung beitragen.

Was PEARL nicht leistet ist die Festlegung komplizierter Datenstrukturen auf Grund der Problemstellung selbst, zusätzlich die Gesamtproblematik der Generierungs- und Bedienungsebene.

Benutzerfreundliche Bedienung für die Plottersoftware

E.Bachner, Kernforschungsanlage Jülich / ZAT

1. Beschreibung der Plottersoftware von Siemens

Die Plottersoftware ermöglicht graphische Darstellungen jeglicher Art auf Plottern, X-Y-Schreibern oder graphischen Sichtgeräten. Die Bearbeitung aller notwendigen Funktionen für Bildaufbau und -ausgabe übernehmen Assembler-Unterprogramme, die zu dem Unterprogramm-Paket PLOT 320/330 zusammengefaßt sind. Für FORTRAN-Benutzer übernimmt die LIBRARY PLTT u.a. die formale Umsetzung der Unterprogramme zu SUBROUTINEN.
Da mit diesen Grundbausteinen jegliche Art von graphischer Darstellung vorgenommen werden kann, ist die Anzahl der Parameter sehr groß (ca. 70). Über jeden dieser Parameter muß man sehr genau Bescheid wissen, ebenso wie über die Software-Struktur. Das bedeutet aber für den, der diese Software zum erstenmal benutzt, das eingehende Studium einer guten Software-Beschreibung. Ist diese nicht vorhanden, muß man sich die notwendigen Erfahrungen und Erkenntnisse durch Probieren aneignen. Wir mußten leider den zweiten Weg gehen.
Nun ist es durchaus möglich, die Software für bestimmte Anwendungsfälle unter Einschränkung der vorhandenen Möglichkeiten zu vereinfachen. Diesen Weg ist Siemens durch Lieferung verschiedener FORTRAN-Bibliotheken gegangen. Sie bieten übergeordnete Funktionen, die die wünschenswerten Vereinfachungen für einen bestimmten Anwendungsfall bringen.
Ich möchte hier näher eingehen auf die LIBRARY DIAG, da wir darauf unser Konzept aufgebaut haben.
LIBRARY DIAG liefert die Hilfsmittel, um zweidimensionale Meßwerte-Diagramme zu zeichnen. Es gibt nur noch drei SUBROUTINEN: zum Zeichnen von

- Achsen
- Kurven
- und zur Textausgabe.

Die Parameter zu jedem dieser Unterprogramme stehen in einer Steuerliste von 50 Worten Länge, d.h. es gibt auch 3 Arten von Steuerlisten.
Ein Beispiel: Ein Diagramm aus einer X- und einer Y-Achse, 2 Textzeilen und 4 Kurven benötigt 8 Listen:

2 Achssteuerlisten
2 Textsteuerlisten und
4 Kurvensteuerlisten.

Durch diese Strukturierung wird die Anwendung sehr viel übersichtlicher und einfacher.
Diese Steuerlisten werden von einem mitgelieferten Programm in einer Steuerlistendatei hinterlegt. Zur Ausführung muß jeweils die erste Steuerliste jeder Art auf eine der ersten 3 Blöcke der Steuerlistendatei transferiert werden. Mit einem weiteren Programm kann man sich Steuerlisten protokollieren lassen. Diese Vereinfachungen durch LIBRARY DIAG sind für unsere Verhältnisse noch nicht weitgehend genug gewesen. Denn noch immer muß der Benutzer über 46 Parameter genau Bescheid wissen, vor allem auch über solche, deren Sinn nicht ohne weiteres einzusehen ist.
Das Ziel unserer Softwareentwicklung ist es, die Programme so zu schreiben, daß die Experimentatoren und Betreuer von Experimenten weitgehend ihre Experimente selbst vom Rechner bedienen können und nur in Ausnahmefällen auf unsere Hilfe angewiesen sind. Diese Art hat sich bei den speziellen Experimentprogrammen bewährt, und das wollten wir auch für die Plottersoftware erreichen.

Konzept

Um dieses Ziel zu erreichen, sollten soviele Parameter wie möglich festgelegt werden bzw. sich aus anderen Parametern errechnen. Das geht natürlich nur unter Einschränkung der Möglichkeiten, die LIBRARY DIAG bietet. Um trotzdem auch für komplexere Diagramme gerüstet zu sein, gibt es weitere Bedienungsebenen, um den ganzen Umfang von LIB.DIAG ausnutzen zu können.

Ausführung

Wie schon beschrieben, besteht bei LIB.DIAG, die Möglichkeit, beliebig viele Steuerlisten von beliebigen Diagrammen in der Steuerlistendatei zu hinterlegen. Allerdings muß der Benutzer selbst über den Platz in der Steuerlistendatei Buch führen, und wo die einzelnen Listen zu den Diagrammen stehen. Diese Funktion wurde von uns erweitert. Es wurden zwei Listen eingeführt:

1. Die Blockbelegungsliste (BBL) ist eine Bitliste, in der die Belegung der Blöcke der Datei durch Steuerlisten vermerkt wird.
2. Die Diagrammliste (DIALI), deren Element 3 Worte lang ist. Jedes Diagramm bekommt eine Nummer zur Identifikation. In dieser Liste sind die Blocknummern der jeweils 1.Steuerliste jeder Steuerlistenart für jede Diagrammnummer eingetragen.

Durch diese Maßnahme konnte auf relativ einfache Weise erreicht werden, daß sich der Benutzer keine Gedanken über freie Blöcke machen muß. Auch alle weiteren Manipulationen erfolgen über die Diagrammnummer und evt. eine relative Steuerlistennummer.

Bedienungen 1.Stufe

Die Bedienungen der 1.Stufe sind für Anwender bestimmt, die praktisch keine Ahnung von der Plottersoftware haben. Zuerst müssen die Listen für ein Diagramm eingerichtet werden. Das geschieht mit der Bedienungsanweisung:

BSEINR; vom BlattSchreiber EINRichten

Dabei wird die nächste freie Diagrammnummer gesucht und ausgedruckt. Außerdem wird in der Bitliste nach freien Blöcken für die Steuerlisten gesucht. Im folgenden Dialog werden folgende Parameter angefragt:

Für maximal 3 Achsen (eine X-Achse und 2 Y-Achsen)

1. Anfangswert
2. Endwert
3. Skalendifferenz
4. Achsbeschriftung (Text)
5. Dateiname(n) der Meßwerte

außerdem

6. Text für Bildunterschrift
7. Text für Bildüberschrift.

Alle weiteren Parameter sind festgelegt bzw. werden aus den angegebenen berechnet.
Als Erweiterung und zusätzlicher Komfort ist geplant, die Achskalierung auf Wunsch aus den Meßwerten selbst berechnen zu lassen.

Damit sind die Parameter für das Normdiagramm festgelegt (siehe Abb.). Mit

PROT: diagnr;

werden die Parameter für dieses Diagramm protokolliert. Die Zeichnungsausgabe wird durch

PLOT: diagnr;

angestoßen. Dabei wird der letzte noch fehlende Parameter ermittelt. Die X-Werte-Datei wird auf ein Endkriterium untersucht, daß vom Sortierprogramm eingetragen wurde. Die dadurch ermittelte Anzahl der Meßwerte wird in die Kurvensteuerliste eingetragen.
Wird ein Diagramm nicht mehr gebraucht, kann es mit

LOE: diagnr;

gelöscht werden, d.h. die entsprechenden Listenplätze werden wieder freigegeben.

Bedienungen 2.Stufe

Während für die Bedienungen der 1.Stufe praktisch keine Kenntnisse über die Struktur der Plottersoftware erforderlich sind, muß man bei der 2.Stufe wenigstens die Steuerlisten kennen.
Soll ein Diagramm erweitert werden, z.B. eine weitere Kurve gezeichnet werden, so ist das möglich mit der Funktion

ERG: diagnr {A T K} ; ERGänzen.

Hierbei wird eine weitere Steuerliste eingeführt, deren Parameter durch einen Dialog ermittelt werden.
Ebenso ist es auch möglich, eine Steuerliste zu streichen mit

LOE: diagnr {A T K} nr;

Will man die Achsskalierung, z.B. dem Experimentfortschritt (Zeit) anpassen oder nur einen Ausschnitt aus seinen Meßwerten gezeichnet haben, kann man mit

AXKORR: diagnr {A T K} nr;

die Achsenparameter nach den augenblicklichen Gegebenheiten ändern.

Bedienungen 3.Stufe

Mit den Bedienungen der 3.Stufe kann man alle Möglichkeiten, die LIBRARY DIAG bietet, ausnutzen. Dazu ist selbstverständlich die genaue Kenntnis der Parameter und der Struktur der Software notwendig.
Mit einem Programm von LIB.DIAG können Steuerlisten von Lochkarten in angebbare Blöcke der Steuerlisten-Datei geschrieben werden. Diese Funktion wurde erweitert.

Mit

LKEINR: anz; von LochKarten EINrichten

werden anz-Steuerlisten eingelesen. Es wird wieder, wie bei BSEINR: eine Diagrammnummer ermittelt und ausgedruckt. Die Platzverteilung in der Steuerlistendatei übernimmt wieder das Programm. Außerdem kann jeder Parameter in den Steuerlisten geändert oder eingegeben werden. Mit

KORR: diagnr $\left\{\begin{matrix}A\\T\\K\end{matrix}\right\}$ nr;

wird eine bestimmte Liste ausgewählt.
Anschließend kann mit dem Parameternamen als Codewort der Zahlenwert oder Text eingegeben werden, z.B.

NCHR = 27;

Anzahl der Schriftzeichen

KORR;

ohne Parameter beendet die Korrektur der Liste, die anschliessend mit den neuen Parametern protokolliert wird.
Das Programm ist weitgehend in FORTRAN geschrieben. Einige SUBROUTINEN wurden ganz oder teilweise in Assembler programmiert:

1. Die Bearbeitung der Bitliste wäre in FORTRAN sehr aufwendig.

2. Die Bearbeitung von Codewortlisten und Bedienungseingaben sind ebenfalls in Assembler einfacher, vor allem deswegen, weil dafür selbstgeschriebene Unterprogramme existieren.

3. Für Zahleneingaben über den Blattschreiber ist die Formatsteuerung von FORTRAN ungünstig. Hier besteht eine elegante Möglichkeit durch das Siemens Unterprogramm FGEK, das die wichtigsten Zahlendarstellungen umcodieren kann.

Das beschriebene Bedienungsprogramm ist erst teilweise fertig. Aus der praktischen Arbeit damit können sich durchaus noch Änderungen am Konzept ergeben.

Eine Ein- und Ausgabe-Steuerung zur Vereinfachung der Prozeßeinheit 3600 für Siemens-Prozeßrechner 310 bis 340

Hans-Joachim Schuster, Physikalisch-Technische Bundesanstalt, Braunschweig

1. Einleitung

Obwohl die mit Rechnern zu führenden Meßprozesse in der Physikalisch-Technischen Bundesanstalt außerordentlich unterschiedlich sind - die PTB besteht aus über 100 Laboratorien der verschiedensten Fachrichtungen - wird für ihren Hardware-Anschluß an Siemens-Prozeßrechner 330 immer nahezu das gleiche Interface benötigt. Von DMA-Anschlüssen abgesehen, besteht es an jedem Rechner aus der Bereitstellung von 4 mal jeweils 8 Stück 16-bit-Digitaleingängen, 4 Stück Digitalausgängen und 16 Alarmeingängen an 4 verschiedenen, im Mittel 50 m vom Rechner entfernten Orten. Mit diesen digitalen Schnittstellen soll in der PTB der Rechnerbereich oder das Interface enden. Alle analogen Bausteine werden, weil sie sehr aufgabenspezifisch sind, der Experimentelektronik zugeordnet.

Diese im Prinzip einfache Interface-Forderung läßt sich mit den Geräten der Prozeßeinheit 3600 [1, 2]) der Firma Siemens leider nur mit sehr großem elektronischen Aufwand realisieren. Dadurch steigen die Interface-Investitionskosten in eine nicht mehr zu vertretende Größenordnung. Sie betragen pro Rechner ca. 160 000.- DM und für alle geplanten 22 Rechner ca. 3,6 Mill. DM, ein Betrag, der dafür nicht zur Verfügung steht. Außerdem fallen noch die Wartungskosten in gleicher Höhe an, wenn eine mittlere Einsatzzeit von 10 Jahren zugrundegelegt wird.

Deshalb wurde eine Ein-Ausgabe-Steuerung als kostensparende Ergänzung zur Prozeßeinheit 3600 oder auch als eigenständige Prozeßeinheit entwickelt, die von einer Münchener Firma in 100 Exemplaren exklusiv für die PTB gebaut wird. Durch diese

EA-Steuerung können der elektronischen Aufwand für das Interface um etwa den Faktor 5 und die Kosten den Faktor 6 reduziert werden.

Im folgenden wird auf die Anwendung der EA-Steuerung, ihren Aufbau und den theoretischen Hintergrund für die damit erzielte drastische Vereinfachung der Interface-Elektronik eingegangen.

2. Anwendung der EA-Steuerung

Die EA-Steuerung stellt bis zu 26 (Maximalausbau) 25-polige Anschlußbuchsen zum Experimentanschluß an das Siemens-Prozeßrechner-System zur Verfügung.

Jede Anschlußbuchse gestattet den im allgemeinen vollständigen Hardware-Anschluß eines digitalen Ein- oder Ausgabegerätes. Das bedeutet, daß zu jeder Anschlußbuchse nicht nur 16 Leitungen für die Ein- oder Ausgabeinformation führen sondern zusätzlich noch 5 freie Wahlleitungen, die intern programmierbar mit Steuersignalen belegt werden können (Anforderung, Quittung, Modusanzeigen). Dazu werden auf einem internen Programmierfeld alle für einen beliebigen Hardwareanschluß notwendigen Signale über Buchsen bereitgestellt und können nach Bedarf auf die Wahlleitungen zu den Anschlußbuchsen gesteckt werden. So läßt sich sogar der Hardware-Anschluß von IEC-Bus- und CAMAC-Geräten realisieren.

Die EA-Steuerung, die als Alternative zu einer voll mit Prozeßsignalformern bestückten Grundsteuerung zu sehen ist, kann wie diese über eine EA-Anpassung entweder direkt an eine EA-Schnittstelle des Rechners, an beliebige EA-Schnittstellen einer Grundsteuerung oder an Platz 0 einer Erweiterungssteuerung angeschlossen werden.

Für die Realisierung unserer Standard-Interface-Anforderung (4 mal jeweils 8 Stück Digital-Eingänge, 4 Stück 16-bit-Ausgänge und 16 Alarmeingänge) bieten sich als 3 Alternativen an, die sich im Preis erheblich unterscheiden (s. Abb.)

Der Aufwand der Alternative 1 ist kaum zu rechtfertigen. Alternative 2 hat gegenüber 3 die Vorteile, daß die Alarmmultiplexung in der EA-Steuerung entfällt, weil diese der Grundsteuerung mit übertragen werden kann (s.u.) und daß Signalformer von Siemens noch betrieben werden können. Alternative 3 ist die mit Abstand billigste und eine den allgemeinen praktischen Anforderungen sehr gut angepaßte Lösung.

Die EA-Steuerung wird wie die Prozeßeinheit 3600 bei zentraler Initiative mit den Befehlsdoppeln ADA DTE oder ADA DTA angesprochen und meldet sich bei peripherer Initiative mit einer peripheren Organisationsforderung (POA) oder Datenanforderung (PDA). Bis auf einige weggelassene BFA-Befehle besteht völlige Software-Kompatibilität zur Prozeßeinheit 3600.

3. Aufbau der EA-Steuerung

Die EA-Steuerung besitzt eine Grundplatine, auf der sich die Elektronik für die Anschlußstellenadressierung, die Handshake-Steuerung, der Alarmscanner und der Ein-Ausgabe-Bus befinden. Auf den Bus können maximal 13 Eingabe-, Ausgabe- oder Alarmplatinen gesteckt werden.

Eine Eingabeplatine steuert 2 Eingabebuchsen, eine Ausgabeplatine 2 Ausgabebuchsen und eine Alarmplatine eine Alarmbuchse mit 16 Alarmeingängen.

Die 26-poligen Anschlußbuchsen von AMP sind jeweils über Drähte fest mit ihrer Steuerplatine verbunden und sind an die Frontplatte der EA-Steuerung angeschraubt. Alle zur Anschlußbuchse führenden Signale - in der Regel Daten und noch nicht festgelegte Signale - und einige zusätzliche Steuersignale sind außerdem über Einzelkontakte von Elko am oberen Rand der Steuerplatine verfügbar. Die Einzelkontakte aller Steuerplatinen ergeben das interne Programmierfeld,auf dem beliebige Signalkonfigurationen der Anschlußbuchsen zusammengesteckt werden können.

Die EA-Steuerung enthält außerdem eine für den Maximalausbau ausreichende Stromversorgung und zwei Anschlüsse für Siemens-EA-Anpassungen für ihren Anschluß an den Rechner und für den Anschluß

einer weiteren geketteten EA-Steuerung. Das Gehäuse ist ein Spezial-AEC-Rahmen der Firma Knürr.

4. Theoretischer Hintergrund für die Vereinfachung der Interface-Elektronik

Im folgenden sollen die wichtigsten Überlegungen und Maßnahmen erläutert werden, die zur Vereinfachung der EA-Steuerung gegenüber der Grundsteuerung mit entsprechenden Signalformern geführt haben.

Für alle Anschlußstellen einer EA-Steuerung wird ein gemeinsames Handshake-System benutzt und die Adressierung logisch dahinter verlegt - wie bei der Subadressierung von Prozeßsignalformern[1]). Dadurch entfallen die Handshake-Systeme in den Prozeßsignalformern[1]). Verglichen mit einer voll bestückten Grundsteuerung werden 15 von 16 Systemen eingespart. Außerdem werden der Taktgenerator für zentrale Initiative wesentlich vereinfacht sowie die Zeitüberwachung und Simulation von Quittungen bei zentraler Initiative überflüssig. Denn auch bei der programmgesteuerten Anwahl einer Anschlußstelle, die mit keiner oder einer defekten Steuerplatine bestückt ist, läuft die Signalfolge über das gemeinsame Handshake-System, das ja von den Steuerplatinen nicht abhängt, ungestört ab und das darauf wartende Programm kann sich nicht festfahren.

Weiterhin wurde die Alarmbearbeitung wesentlich vereinfacht. Die Grundsteuerung multiplext Alarme von 16 Primäralarmeingängen [2]). Die Alarmnummer, die den Eingang, über den der Alarm eingetroffen ist angibt, wird bei einem Primäralarm mit diesem selbst extern gesteuert an den Rechner übertragen. Mit den Alarmen jedes Primär-Einganges können jeweils das Setzen eines oder auch mehrerer Bits einer 16-bit-Alarmeingabe 3612 [1]) dem Rechner angezeigt und das Abtragen dieser Alarmeingabe veranlaßt werden. Auf diese Weise lassen sich pro Primäralarm-Eingang 16 Sekundäralarm-Eingänge mit zentralgesteuerter Alarmnummernabfrage schaffen. Bei 16 Primäralarm-Eingängen ergeben sich dann maximal 256 Sekundäralarm-Eingänge. Diese große Anzahl wird in der Praxis

sicherlich nie benötigt und könnte auch wegen der begrenzten Rechenkapazität des Rechners schwerlich ausgenutzt werden, obwohl der hohe Aufwand für die 16 Primäralarm-Eingänge - fast die Hälfte der Grundsteuerung - vom Kunden bezahlt werden muß. Andererseits ist aber die Verfügbarkeit nur eines einzigen Primäralarm-Einganges wie bei der Erweiterungssteuerung [2]) von Siemens im allgemeinen zu wenig, weil man dann auf 16 Sekundäralarm-Eingänge beschränkt ist und auch die Möglichkeit der Kettung und des Blocktransfers, die auch je einen Primäralarm-Eingang benötigen, verliert.

Aus diesem Grunde wurden von der entwickelten EA-Steuerung 4 Primäralarm-Eingänge und damit 30 bis 60 Sekundäralarm-Eingänge - je nachdem ob Kettung und Blocktransfer erforderlich sind - angeboten. 4 Primäralarm-Eingänge lassen sich besonders einfach realisieren (s.u.), reichen in der Praxis fast immer aus und stehen auch in einem vernünftigen Verhältnis zur Kapazität des Rechners.

Sekundäralarme sind prinzipiell auch bezüglich ihrer Bearbeitungszeit Primäralarmen nicht unterlegen, weil der Aufwand für das programmgesteuerte Abfragen der Alarmnummer bei Sekundäralarmen durch die erforderlichen Synchronisationsmaßnahmen (Freigabebefehl BFA PAFR) bei Primäralarmen ausgeglichen wird [3]). Daher kann auch die Verkürzung der Bearbeitungszeit kein Grund sein, die Anzahl der Primäralarm-Eingänge zu erhöhen, um dann Sekundäralarme durch Primäralarme ersetzen zu können.

Für die Realisierung der 4 Primäralarm-Eingänge der EA-Steuerung gibt es zwei Möglichkeiten. Wenn die EA-Steuerung an eine Grundsteuerung angeschlossen ist, werden einfach 4 Primäralarm-Eingänge der Grundsteuerung über unbenutzte Signalwege der EA-Anpassung weitergereicht. In diesem Falle entfällt in der EA-Steuerung die gesamte Elektronik für periphere Initiative und die Grundsteuerung wird besser ausgenutzt. Deshalb ist auch die Hardware-Konfiguration 2 auf der Abbildung, in der 4 EA-Steuerungen an eine Grundsteuerung angeschlossen sind, sehr vorteilhaft.

Ist keine Grundsteuerung vorhanden, werden die 4 Alarmeingänge durch einen sehr einfachen Scanner mit etwa einem Neuntel des Aufwandes in der Grundsteuerung realisiert. Durch die Bereitstellung von 4 statt 16 Primäralarm-Eingängen wird der elektronische Aufwand wesentlich stärker als proportional reduziert, weil dann u.a. jeder Primäralarm mit einem eigenen Bit numeriert werden kann - 4 Bits stehen dafür insgesamt zur Verfügung. Dadurch entfällt z.B. die Alarmnummerncodierung vom 1 aus 16 - in den 4-bit-Binärcode.

Weitere Einsparungen ergaben sich durch Weglassen von Kontroll- und Überwachungsfunktionen, die sich entweder durch die Neukonstruktion von selbst erledigten, wie die Zeitüberwachung bei zentraler Initiative (s.o.), oder deren Nutzen bezüglich der Verbesserung der Verfügbarkeit nicht klar erwiesen ist. Denn Kontrollelektronik kann die Verfügbarkeit durchaus auch vermindern, weil sie einerseits zwar Fehler schneller finden hilft, aber andererseits auch defekt werden kann und so zu neuen Fehlern führt. Viele aufwendigen Kontrollmaßnahmen des Prozeßelementes 3600 werden z.B. im CAMAC- oder IEC-Bus-System nicht durchgeführt.

Weiterhin sind auch wichtige Vereinfachungen und Verbesserungen an der Prozeßsignalformung und beim "Blocktransfer" vorgenommen worden, auf die jetzt im einzelnen nicht weiter eingegangen werden kann. Statt dessen sollen die wichtigsten Ergebnisse dieser Arbeit noch einmal zusammengefaßt werden.

Durch die angedeuteten Vereinfachungen und Verbesserungen konnten der elektronische Aufwand und die Kosten für ein Standard-Interface für Siemens-Prozeßrechner 310-340 um etwa den Faktor 5 reduziert werden.

Trotz dieser drastischen Einsparungen ergeben sich keinerlei Einschränkungen oder Nachteile im praktischen Gebrauch. Es konnten im Gegenteil noch wichtige Verbesserungen erzielt werden, wie z.B. die interne Programmierbarkeit, die Bereitstellung häufig benötigter Steuersignale und Register, für den Geräteanschluß, verbesserte Alarmbearbeitung[3]), höhere Geschwindigkeit und TTL-Pegel bei der Prozeßsignalformung.

Abschließend sei noch betont, daß die EA-Steuerung ursprünglich nicht als Konkurrenz-Interface zur Prozeßeinheit 3600 konzipiert wurde, sondern als Kostensparende hard- und softwarekompatible Ergänzung dazu. Später stellte sich aber heraus, daß sie auch sehr gut als eigenständiges Interface-System betrieben werden kann.

Literaturangaben

[1]) Siemens Systeme 300 - 16 bit,
Prozeßeinheit 3600 Prozeßsignalformer,
Beschreibung Bestell-Nr. ESTE 4 - 108a/000

[2]) Siemens Systeme 300 - 320/330,
Grundsteuerung 3601 Erweiterungssteuerung 3602,
Beschreibung Bestell-Nr. E AUT 3-107

[3]) H.-J. Schuster: Überlegungen zur Behandlung peripherer Anforderungen in Interface-Systemen,
PTB-Bericht in Vorbereitung.

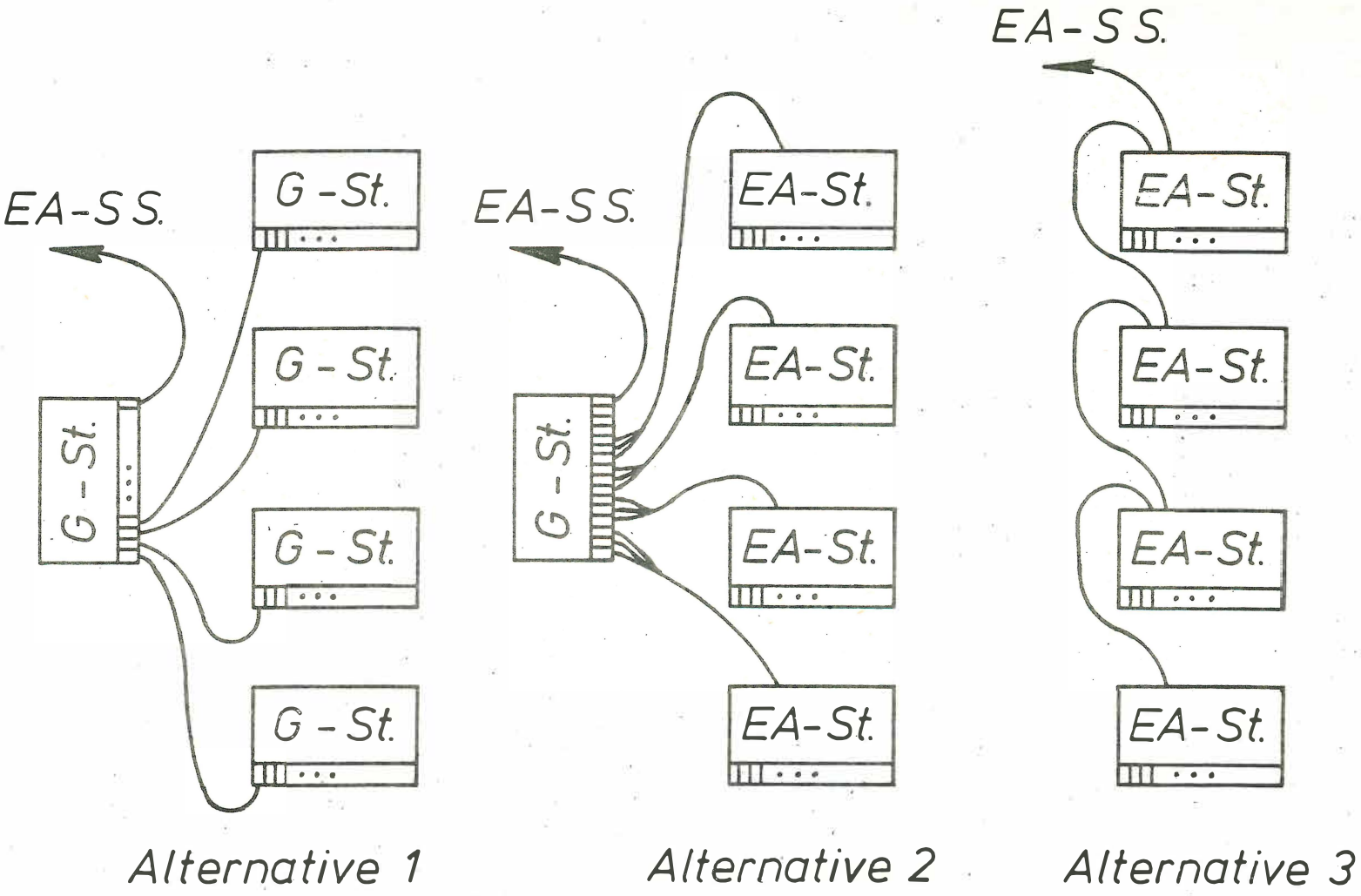


Abb.: 3 Alternativen zur Realisierung der Standard-Interface Anforderung in der PTB

Kurvensichtgerät für eine DVA 301

A. John *

1. Kurvensichtgerät in der Prozeßdatenverarbeitung

Bei großen Anlagen, zumal wenn sie mit Hilfe eines Prozeßrechners betrieben werden, geschieht die Überwachung durch den Menschen meist von einer zentralen Warte aus. Zeigerinstrumente, Kontrollampen und Meßprotokolle geben viele Einzelinformationen, aus denen mühsam ein umfassendes Bild der Anlage zusammengesetzt werden muß. Um eine Kontrolle ausüben zu können, müssen die wichtigsten Prozeßgrößen in übersichtlicher Darstellung zugänglich sein. Physikalische Größen als Funktionen der Zeit werden am leichtesten erfaßbar in einer dem Ingenieur geläufigen Weise dargestellt,in Form von Diagrammen, Kurven oder Kennlinien. Die graphische Darstellung birgt auf dem gleichen Raum eine wesentlich höhere Informationsdichte, die zudem schneller erfaßbar ist, als eine rein alphanumerische Darstellung (z.B. in Tabellenform). Die Tendenz zur geräusch- und papierlosen Nachrichtenübermittlung führt zwangsläufig zur Darstellung auf einem Bildschirm. Da mehrere Kurven dargestellt werden sollen, die gut unterscheidbar sein müssen, wurde als Ausgabemedium ein Farbfernsehgerät gewählt.

Da die Kurven allein noch keine Aussagekraft haben, kann auch alphanumerischer Text auf dem gesamten Bildschirm ausgegeben werden (alphanumerisches Sichtgerät).

---

* Dipl.-Ing.A.John ist wissenschaftlicher Assistent am Lehrstuhl und Institut für Allgemeine Elektrotechnik und Automatisierung der RWTH-Aachen, Direktor Professor Dr.-Ing. Hans Henning.

Das Kurvensichtgerät wurde im Rahmen einer 6-monatigen Diplomarbeit von Herrn F.Walbrodt am Institut gebaut. Das Gerät wurde vom Konzept her an die Kurvensichtstation KUSA 300 der Firma Siemens angelehnt, der wir für die zur Verfügung gestellten technischen Unterlagen unseren Dank aussprechen möchten.

2. Aufbau der Bildschirmdarstellung

In Bild 1 zeigt den Bildschirminhalt, wobei im Kurvenfeld nur zwei der vier möglichen Kurven dargestellt sind.

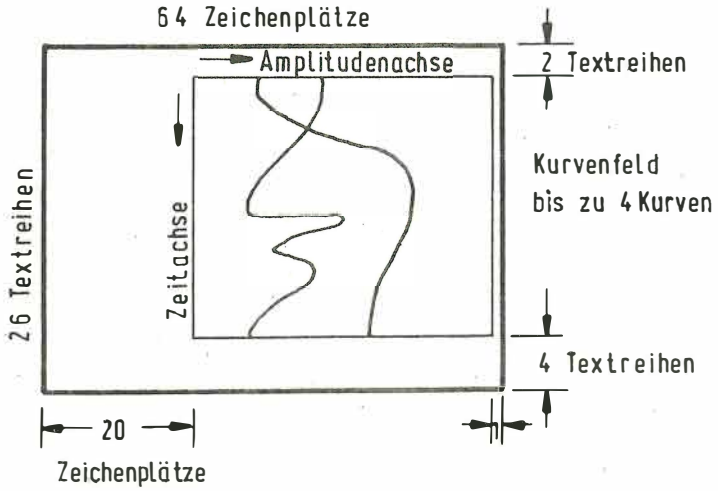


Bild 1. Aufteilung des Bildschirmes

Das Kurvenfeld

Im Kurvenfeld können maximal vier farblich unterscheidbare Kurven in Form eines Liniendiagramms dargestellt werden. Jede Kurve hat 200 Meßwerte in der Abszisse und eine Auflösung von 8 bit, d.h. 256 Amplitudenstufen (Ordinate). Die Kurven sind gegenüber mathematischer Darstellung um 90^{o} gedreht, bedingt durch das Fernsehprinzip des Monitors.

Das Textfeld

Das Textfeld besteht aus 26 Textreihen mit jeweils 64 alphanumerischen Zeichen mit 6bit-USASCII-Zeichenvorrat. Die Texteingabe ist per Software frei über den Bildschirm positionierbar. Die maximale Datenübertragungsrate an der geänderten ZAP 301 beträgt 5 kHz; das bedeutet, daß ein Bild in etwa einer Drittel Sekunde vollständig mit Text gefüllt werden kann.

Betriebsarten

Wichtigste Betriebsart für die Prozeßdatenverarbeitung ist die schritthaltende Meßwertdarstellung (z.B. Ein neuer Meßwert pro Sekunde). Die Kurve wandert dabei nach oben, der älteste Meßwert geht verloren.

Um bei technisch-naturwissenschaftlichen Problemstellungen Parameterstudien treiben zu können, wählt man die blockweise Ausgabe. Dabei wird das Kurvenfeld mit maximaler Datenübertragungsrate gefüllt.

Kurvendarstellung

Innerhalb des Kurvenfeldes sind verschiedene Darstellungsarten der Kurvenzüge möglich:

- Es werden nur die Original-Meßpunkte dargestellt.
- Die Meßpunkte werden durch Linien miteinander verbunden.
  (Wichtig bei stochastisch auftretenden Meßwerten, siehe Bild 2).

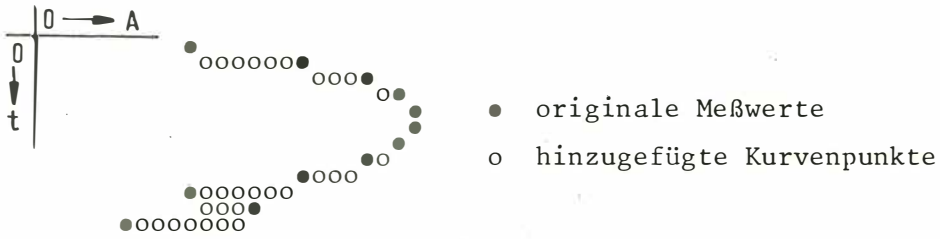
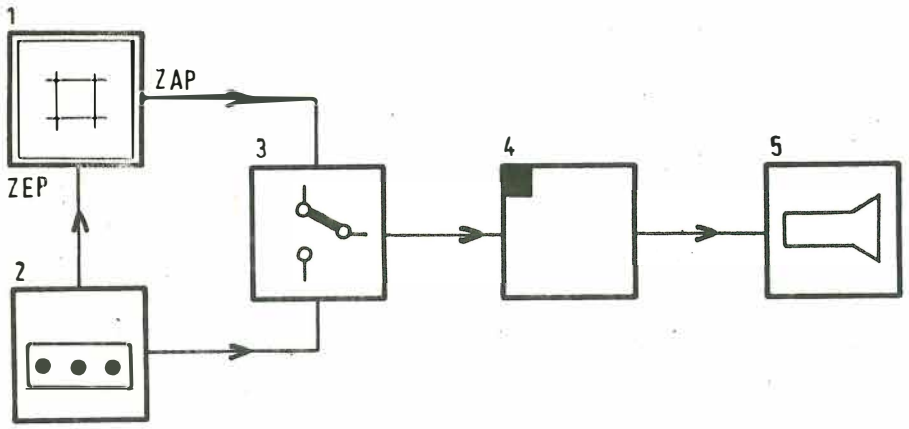


Bild 2. Verbindungspunkte zwischen Meßwerten

- Die Kurve wird im Profil dargestellt, d.h. die Kurve wird unterhalb der Meßpunkte bis zur Nullinie hellgetastet.
- Das Kurvenfeld wird durch unterlegte Hilfslinien mit veränderlicher Rasterung herausgehoben.
- Unipolare und bipolare Darstellung (Nullinie der Amplitudenachse in Kurvenfeldmitte) sind möglich.
- Das Kurvenfeld kann durch Trimmer über den Bildschirm verschoben werden.
- Das Kurvenfeld kann völlig unterdrückt werden.

3. Funktionale Gliederung des Aufbaus

Bild 3 zeigt einen einfachen Übersichtsschaltplan des Kurvensichtgerätes. Es ist über eine ZEP/ZAP-Schnittstelle an den Rechner angekoppelt.



1 Rechner (PR 301)
2 Tastatur (ASCII-Vorrat und Sonderfunktionen)
3 Multiplexer (Rechner/Tastatur)
4 Steuerung mit Bildwiederholspeicher
5 Monitor (Farbfernseher)

Bild 3. Übersichtsschaltplan des Kurvensichtgerätes

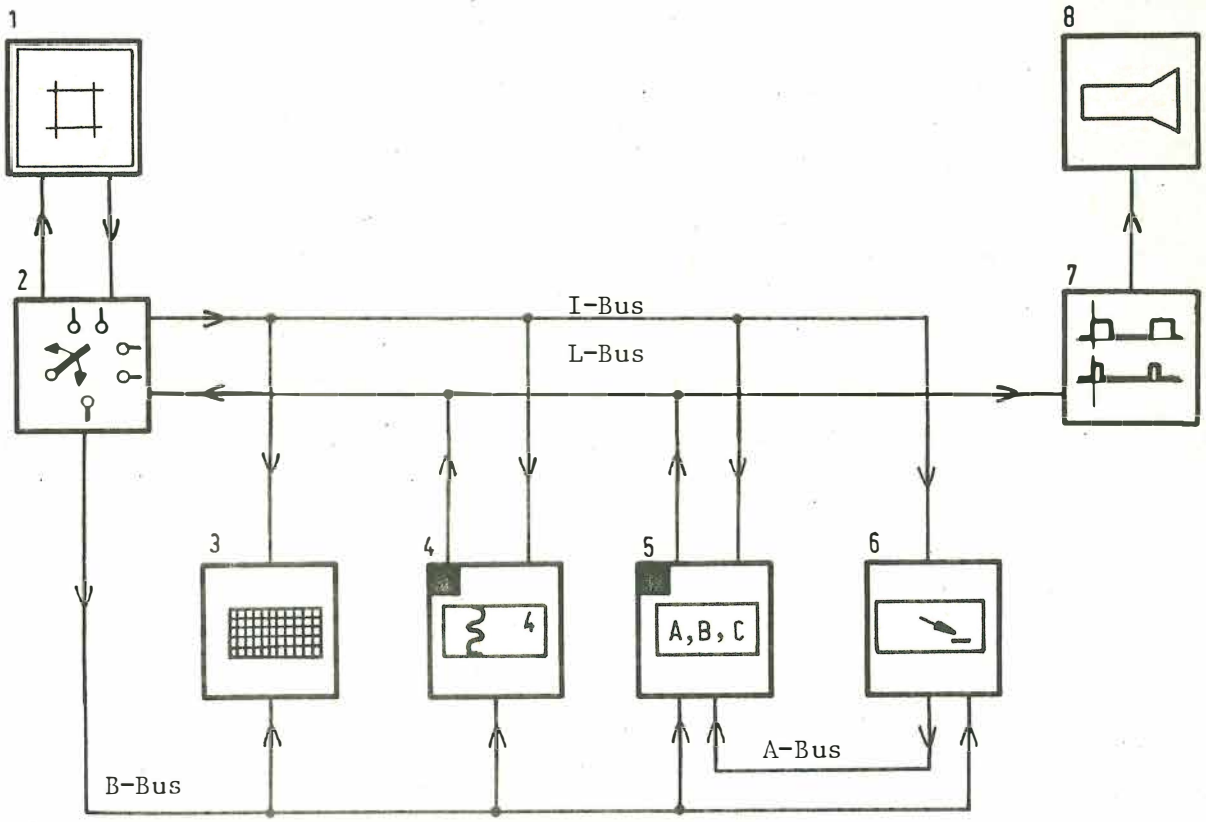
Die Tastatur, über die Daten wahlweise in den Rechner oder den Bildspeicher eingegeben werden können, ist vorläufig nur geplant. In Betrieb genommen wurde das Ausgabegerät, dessen innere funktionale Gliederung Bild 4 zeigt.

Die einzelnen Blöcke, die auch gerätemäßig meist auf einer Platine realisiert wurden, sind Steuerwerk, Rastergenerator, 4-Kurven Speicher, Textspeicher und Blinkerzähler.

Die einzelnen Blöcke sind durch Bussysteme miteinander verbunden:

- I-Bus Informationsbus (Dateneingang)
- L-Bus Lesebus (Zeichen und Kurven)
- A-Bus Adress-Bus (Textadresse)
- B-Bus Befehls-Bus (Steuerung)

Durch diesen Aufbau ist das Gerät universell verwendbar und unterliegt keinen Einschränkungen z.B. hinsichtlich der Textausgabe wie die KUSA 300 der Firma Siemens.



1 Rechner (PR 301)
2 Steuerwerk
3 Rastergenerator
4 Kurvenspeicher für 4 Kurven
5 Textspeicher
} Bildwiederholspeicher
6 Blinkerzähler
7 Synchron-Signalerzeugung
8 Monitor (Farbfernseher)

Bild 4. Funktionale Gliederung des Kurvensichtgerätes

4. Codierung

Die ZAP 301 bietet die Möglichkeit, 8 bit plus 1 Steuerbit parallel auszugeben. Diese 9 bit werden nach Bild 5 zum Verschlüsseln der verschiedenen Informationen benutzt.

AST1	ASPUR8	7	6	5	4	3	2	ASPUR1	
0	0	0	0	x	x	x	x	x	Befehlsworte
0	0	0	1	x	x	x	x	x	ASCII-Zeichen (Text)
0	0	1	0	x	x	x	x	x	ASCII-Zeichen (Text)
0	0	1	1	x	x	x	x	x	Befehlsworte
1	x	x	x	x	x	x	x	x	8-bit-Information (Meßwerte, Zustands-information, Blinkeradressen)

Bild 5. Codierung der Datenwörter

Es gibt vier verschiedene Datenworte, die im wesentlichen durch die ersten 4 bit (s. Bild 5) gekennzeichnet werden:

0000-Worte sind Mehrwortbefehle

- Farbwahl der Texteingabe
- Auswahl des Kurvenspeichers (1 bis 4)
- Adressierung des Blinkerzählers (Adresse folgt)
- Anwahl des Kurven-Zustandsregisters (Zustand folgt)
- Anwahl des Raster-Zustandsregisters (Zustand folgt)

0011-Worte sind Direktbefehle

- Blinkersteuerung (7)
- Löschbefehle (7)
- ROLL-UP

0001- und 0010-Worte beinhalten den USASCII-Code

1xxx-Wort sind 8-bit-Informationsworte

- Meßwerte
- Zustandsinformationen
- Blinkeradressen

5. Versorgung mit Software

Das Organisationsprogramm ORG-A 301 enthält einen Zusatz, der einerseits den Matrixdrucker (ZAP) über ZAP-Aufrufe, andererseits 4 Sichtgeräte (ZEP/ZAP) über Blattschreiberaufrufe (Blattschreibernummer >1) versorgen kann. Bei der Textausgabe wird für den Matrixdrucker und die Sichtgeräte die gleiche CODE-Tabelle (Intern Code/ASCII) verwendet. Alle übrigen Ausgaben müssen durch Anwenderprogramme und ein Makro-Programmpaket speziell für die Kurvensichtstation erfolgen. Das zugehörige Programm-Paket ist eng an das Programm-Paket MAKUSAO der Firma Siemens angelehnt und enthält folgende Makro-Aufrufe, die zur Textausgabe, Rastereinstellung, Kurven-Zustandsänderung und Meßwertausgabe dienen:

Zur Textausgabe

MA BILO Bild löschen, Blinker an den Bildanfang

MA TEXT = FARBE,ADR
Textausgabe von Adresse ADR bis Endezeichen in der angegebenen Farbe. Bei FARBE = Ø gilt die zuletzt gewählte Farbe.

MA TPOS = FARBE,REIHE,PLATZ,ADR
Freipositionierte Textausgabe von ADR bis Endezeichen in der angegebenen Farbe, auf den Bildschirm geschrieben von der in REIHE und PLATZ bezeichneten Stelle an.

Zur Rastereinstellung

MA RAST = FARBE,MEART,AMPLI,ZEITA,PUNKT
Einblenden des Kurvenfeldes mit einer gewählten Farbe, einer Meßart, einer Amplituden-Achseneinteilung, einem Zeitachsenraster und einem weiteren Rasterparameter.

MA KFEI = EIN
Kurvenfeld ein- oder ausblenden

Zur Kurven-Zustandsänderung

MA KEIN = KNR,EIN,DARST
Einblenden der Kurve KNR mit Darstellungsart DARST

MA KFAR = KNR,FARBE
Die Farbe der Kurve KNR verändern

MA KBLI = KNR,EIN
Das Blinken der Kurve KNR ein- oder ausschalten

MA KPRO = KNR,EIN

Profildarstellung von KNR ein- oder ausschalten

MA KLIN = KNR,AUS

Verbindungslinien zwischen den Meßwerten einer Kurve KNR unterdrücken oder nicht

MA KLOE = KNR

Die Kurve KNR löschen; bei KNR = 4 alle Kurven löschen

Zur Meßwertausgabe

MA MAUS = KNR,Z,ADR

Ausgabe einer Anzahl Z von Meßwerten an den Kurvenspeicher KNR, auszulesen von ADR an

6. Schaltungs- und Geräteaufwand

a) ZAP/ZEP Ein-/Ausgabegeschwindigkeit auf ca. 10 kHz maximal erhöhen (4 Kondensatoren)

b) 1 Farbfernseher, ca. 2000,- DM

Notwendige Eingriffe und Ergänzungen:

- o BAS-Eingang (2 Widerstände, 1 Kondensator)
  Ankopplung über 75-Ω-Koax-Kabel
- o RGB-Eingang
  Ankopplung über drei 75-Ω-Koax-Kabel
  Für Entfernungen bis 5 m mit RGB-Pegel.
  Für Entfernungen bis 15 m mit TTL-Pegel über Leitungstreiber und -empfänger.

c) Steuerung

Ein 19-Zoll-Rahmen mit 2 Netzteil-Platinen
4 Kurvenspeicher-Platinen (modular)
1 Steuerwerk-Platine
1 Synchronsignalerzeugung-Platine
1 Blinkeradreßzähler-Platine
1 Textspeicher-Platine
2 Rastergenerator-Platinen

Materialkosten: ca. 1500,- DM

Arbeitszeit für den Nachbau nach Unterlagen: ca. 3 Mann-Monate

Cross-Assembler für den Mikroprozessor M6800

W. Kristen *

1. Ausgangslage

Am Institut für Allgemeine Elektrotechnik und Automatisierung (IAEA) der RWTH-Aachen sollte mit dem Mikroprocessor M6800 der Firma MOTOROLA ein Mikrocomputer für Steuer- und Regelzwecke aufgebaut werden. Dieser Mikroprozessor wurde ausgewählt, weil er nur eine Versorgungsspannung (+5V) benötigt, einen großen und leistungsfähigen Befehlsvorrat aufweist und nicht zuletzt, weil ein anderes Institut uns den als Quellsprache-Kartenstapel vorliegenden, in FORTRAN geschriebenen Cross-Assembler überlies.

2. Assembler und Cross-Assembler

Die Programmierung eines Rechners kann in Maschinensprache, einer maschinenorientierten Programmiersprache oder in einer problemorientierten Programmiersprache erfolgen. Bei der Programmierung in einer maschinenorientierten Sprache wird ein Programm zur Übersetzung in die Maschinensprache, der sogenannte Assembler benötigt. Erfolgt die Übersetzung nicht auf der Rechenanlage, für die der Maschinencode erzeugt wird, sondern auf einem ganz anderen Computer, so wird das Übersetzerprogramm Cross-Assembler genannt. Um weitgehend anlagenunabhängig zu sein, werden Cross-Assembler im allgemeinen in einer höheren Programmiersprache (z.B. in FORTRAN) geschrieben.

Eine, in einer maschinenorientierten Sprache geschriebene Anweisung ist im allgemeinen in folgende Felder aufgeteilt, wobei nicht alle Felder besetzt sein müssen:

Symbolische Adresse (Marke)	Befehl	Operand	Kommentar

Die Aufgabe des Assemblers besteht darin, jede Anweisung sequentiell abzuarbeiten, auf formale Richtigkeit zu überprüfen und zu übersetzen. Falls eine Marke vorhanden ist, muß sie zusammen mit dem Stand des Programmzählers in eine Sym-

---

* Dipl.-Ing.W.Kristen ist wissenschaftlicher Mitarbeiter am Lehrstuhl und Institut für Allgemeine Elektrotechnik und Automatisierung der RWTH-Aachen, Direktor Professor Dr.-Ing. Hans Henning.

Der Cross-Assembler wurde im Rahmen einer Diplomarbeit von Herrn J.Weskott auf der Rechenanlage des Institutes [1], einer DVA 301 mit Plattenspeicher, Kartenleser, Lochstreifenstanzer, Matrixdrucker und Blattschreiber, ablauffähig gemacht.

boltabelle eingetragen werden. Bei dem Befehl muß untersucht werden, ob es sich um einen Maschinenbefehl oder eine Anweisung für den Assembler (Pseudo-Anweisung) handelt. Aus dem Operadenteil müssen die angesprochenen Register und die Adressen der für diesen Befehl benötigten Speicherplätze ermittelt werden. Da im Operanden auch eine erst später definierte symbolische Adresse stehen kann, der noch kein Programmzählerstand zugewiesen ist, werden Assembler üblicherweise für zwei Programm-Durchläufe ausgelegt. Der allgemeine Programmablauf für den ersten Durchlauf kann aus Bild 1, der für den zweiten Durchlauf aus Bild 2 entnommen werden [2].

3. Ursprüngliche Version des Cross-Assemblers

Die dem Institut überlassene Version des Cross-Assemblers war für eine Rechenanlage SIGMA-9 der Fa. RANK-XEROX mit einer Wortlänge von 32 bit bestimmt. Das Programm bestand aus dem Hauptprogramm und 34 Unterprogrammen. Außerdem enthielt das Programm Aufrufe für eine Code-Transformation und für die Funktionen 'Logical AND' und 'Logical OR' zweier INTEGER-Zahlen. Diese Funktionen waren offensichtlich in das Betriebssystem der SIGMA-9 eingebaut.

Der Cross-Assembler arbeitete in drei Durchläufen, wobei der dritte Durchlauf nur zum wahlweisen Ausdrucken der Symboltabelle benötigt wurde. Die Eingabe der Quellsprache erfolgte von einem Externspeicher (Magnetband) im EBCDIC-Code. Für den Übersetzungsvorgang wurden die Zeichen in den ASCII-Code transformiert und vor der Ausgabe wieder in den EBCDIC-Code zurücktransformiert.

Eine Übersetzung der einzelnen Programmteile ohne Binden ergab eine Gesamtlänge von 10700 Worten. Durch ein Binden wären noch ca. 3500 Worte hinzugekommen. Da auf der Rechenanlage des Institutes nur ein Laufbereich von knapp 10000 Worten zur Verfügung steht, war das Programm in dieser Form nicht ablauffähig.

4. Analyse des Cross-Assemblers

Um Änderungen an dem Cross-Assembler vornehmen zu können, mußten sein Aufbau und die Funktion der einzelnen Unterprogramme untersucht werden. Als Hilfsmittel stand hierzu nur das Programming-Manual der Fa. MOTOROLA [3] zur Verfügung, in dem u.a. der Aufbau der Programmiersprache und die Funktionen des Assemblers beschrieben werden.

Die einzelnen Unterprogramme wurden in insgesamt elf Gruppen aufgeteilt. Innerhalb einer Gruppe kamen nur Unterprogramme vor, die einander nicht aufriefen. Die Gruppen selber waren hierarchisch angeordnet. Ein in einer Gruppe stehendes

Unterprogramm konnte nur aus einer übergeordneten Gruppe aufgerufen werden und nur Unterprogramme aus tiefer liegenden Gruppen aufrufen. Die unterste Gruppe enthielt nur solche Programme, die keine anderen Programme mehr aufrief (s. Bild 3).

Bei der Analyse des Cross-Assemblers wurde bei den zur untersten Gruppe gehörenden Programmen begonnen und dann zur nächst höheren Gruppe übergegangen. Diese Analyse nahm etwa ein Drittel der Gesamtbearbeitungszeit in Anspruch.

5. Erforderliche Änderungen

Code-Änderungen

Da in der DVA 301 ein alphanumerisches Zeichen in einem 6 bit breiten Interncode dargestellt wird, bei der SIGMA-9 aber der 8 bit breite EBCDIC-Code verwendet wird, bereitete einerseits der Übergang von der Wortlänge von 32 bit auf die Wortlänge von 24 bit der DVA 301 wenig Schwierigkeiten, machte aber andererseits das Überarbeiten der beim Formatwechsel und der Code-Transformation benötigten Programmteile erforderlich. Durch Verwenden einer einzigen Code-Tabelle für Hin- und Rücktransformation konnte zusätzlich Speicherplatz eingespart werden.

Prozeß-FORTRAN

Das in den FORTRAN-Compiler der Fa. SIEMENS eingebaute Prozeß-FORTRAN ermöglicht ein einfaches Verschieben von Bitmustern und die Ausführung logischer Operationen. Hierdurch konnten einige Unterprogramme vereinfacht und verkürzt werden bzw. ganz entfallen. Auch wurde von der Möglichkeit, PROSA-Zeilen in ein FORTRAN-Programm einbauen zu können, reichlich Gebrauch gemacht.

Speicherplatz-Einsparung

Besonderer Wert wurde auf das Einsparen von Speicherplätzen gelegt, da man hoffte, den Cross-Assembler linear binden zu können. Deshalb wurden alle INTEGER-Variablen als INTEGER*2 vereinbart. Subroutinen, die nur von einem Programm aufgerufen wurden, baute man in das aufrufende Programm ein und sparte so die zur Aufrufvorbereitung, Parameterübergabe und zum Rücksprung benötigten Speicherplätze ein.

Daten-Eingabe und -Ausgabe

Die Eingabe der Quellsprache sollte über den Kartenleser erfolgen. Deshalb wurde in FORTRAN eine Subroutine geschrieben, die die Lochkarten über den Kartenleser las und den Karteninhalt auf dem Plattenspeicher hinterlegte, wo er für die beiden Übersetzer-Durchläufe zur Verfügung stand. Es zeigte sich aber, daß das Einlesen der Lochkarten erschreckend langsam erfolgte. Deshalb wurden die für das Einlesen der Karten und für den Datentransfer mit dem Plattenspeicher benötigten Programmteile in PROSA geschrieben. Auch die Ausgabe des Maschinencodes in ASCII-Zeichen über den Lochstreifenstanzer wurde in PROSA programmiert.

6. Overlay-Binder

Nach den oben beschriebenen Änderungen betrug die Programmlänge, einschließlich COMMON-Bereich, aber ohne Binden etwa 7700 Worte. Nach dem linearen Binden war das Programm mit etwa 11200 Worten wieder zu lang. Es mußte deshalb mit dem Overlay-Binder segmentiert gebunden werden.

Bei einem in Overlay-Struktur gebundenen Programm wird das vollständige Programm auf dem Externspeicher hinterlegt. Nach dem Start steht nur das Rootsegment, bestehend aus dem Bereitstellungsteil, der Segmentwechsel-Routine und dem Hauptprogramm, ständig im Arbeitsspeicher. Alle anderen Programmteile (Segmente) werden erst nach ihrem Aufruf in sogenannte Laufbereiche im Arbeitsspeicher geladen. Nach ihrer Abarbeitung können sie entweder wieder auf den Externspeicher austransferiert werden oder die nachfolgenden Programmteile überschreiben die vorhergehenden. Die Länge eines Laufbereiches wird bestimmt durch den Speicherbedarf des längsten Segmentes, das in ihm ablaufen soll.

Bild 4 erläutert die Struktur und den Platzbedarf eines Overlay-Programms mit einem Laufbereich am Beispiel eines Hauptprogramms A, das die Unterprogramme B, C, D und E aufruft. Die Unterprogramme rufen sich nicht gegenseitig auf. Der auf dem Externspeicher benötigte Speicherplatz entspricht der Programmlänge nach einem linearen Binden.

Es zeigte sich, daß die bei der Analyse des Cross-Assemblers erstellte Programm-Hierarchie genau den Anforderungen des Overlay-Binders entsprach. Durch Zusammenfassen kürzerer Module zu einem, den Speicherplatz eines Laufbereichs optimal ausnutzenden Segment, konnte die Zahl der Segmenttransfers verringert werden. Das gebundene, ablauffähige Programm belegt im Hauptspeicher 8500 Zellen.

7. Testen und Optimieren

Nachdem mit einem kleinen Programm das einwandfreie Arbeiten des in Overlay-Struktur gebundenen Cross-Assemblers überprüft worden war, wurde ein Testprogramm geschrieben, das lediglich alle Befehle des Mikroprozessors in allen Adressierungsarten und einige Assembler-Anweisungen enthielt (etwa 260 Lochkarten). Zu unserem Entsetzen dauerte die vollständige Übersetzung mit Protokoll und Ausgabe des Maschinencodes knapp 41 Minuten. Solche Laufzeiten sind aber typisch für auf einer DVA 301 kompilierte Programme mit zahlreichen Subroutinen. Der Compiler liefert nämlich einen Maschinencode mit dem Befehlsvorrat der DVA 305, der laufend die zeitraubende Befehlssimulation durch das Betriebssystem erforderlich macht. Außerdem werden bei jedem Subroutine-Aufruf einige Standard-Unterprogramme zur Aufruf-Vorbereitung, Parameter-Übergabe und zum Rücksprung durchlaufen.

Deshalb wurden diejenigen Subroutinen, die während dem Übersetzen am häufigsten durchlaufen werden, in PROSA umgeschrieben und teilweise die Aufgaben anderer Subroutinen in sie integriert. Der Maschinencode dieser Subroutinen war im Schnitt um ein Fünftel kürzer als der vom Compiler gelieferte Maschinencode, obwohl weitestgehend nur Befehle der DVA 301 verwendet wurden.

Die Befehlsliste wurde ursprünglich linear durchsucht, um den Befehlscode zu ermitteln. Dies machte bei 123 Listenelementen im Mittel 62 Suchschritte erforderlich, die in beiden Assembler-Durchläufen erfolgen. Da der mnemotechnische Code der Befehle als ASCII-Zeichenfolge in der Befehlsliste steht und die Buchstaben A bis Z im ASCII-Code aufsteigende Werte einnehmen, wurde die Befehlstabelle alphabetisch - also von niederen zu höheren Binärwerten verlaufend - geordnet und ein binäres Suchverfahren programmiert. Damit müssen maximal nur noch sieben Suchschritte erfolgen, bis ein Befehl gefunden ist.

Nach diesen Änderungen benötigt der Cross-Assembler zum Übersetzen des oben erwähnten Testprogramms nur noch 13 Minuten, von denen jeweils 2,5 min auf das Einlesen der Lochkarten und das Drucken des Protokolls entfallen.

8. Aktuelle Version des Cross-Assemblers

Der zur Zeit an unserem Institut verwendete Cross-Assembler hat folgende Eigenschaften:

- o Eingabe der Quellsprache über Lochkarten oder aus einer Datei des Plattenspeichers
- o Eingabe im BCDIC-Code (Locher Ø26) oder EBCDIC-Code (Locher Ø29)

- o Protokoll-Ausgabe auf Schnelldrucker oder Blattschreiber
- o Eingabe eines Datums für die Protokoll-Überschrift möglich
- o Ausdrucken von Adresse, Befehlscode und Operand im Protokoll wahlweise als Oktal-Zahl, Dezimal-Zahl oder Hexadezimal-Zahl
- o Übersetzen mit oder ohne Protokoll
- o Ausdrucken der Symboltabelle auf Anforderung
- o Ausgabe des Maschinencodes auf 8-Kanal-Lochstreifen im ASCII-Code
- o Übersetzen mit oder ohne Ausgabe von Maschinencode
- o Rücksetzen auf voreingestellte Eigenschaften (Eingabe im BCDIC-Code über Kartenleser, Protokoll mit Hexadezimal-Zahlen, Ausgabe von Maschinencode, keine Symboltabelle)

In der urspünglichen Version durfte im Operandenteil keine symbolische Adresse stehen, der erst später ein Wert zugewiesen wurde (Vorwärts-Referenz). In diesen Fällen wurden im 1. Durchlauf 3 Byte Speicherplatz für den Befehl reserviert. Wurde aber im 2. Durchlauf, wenn die Symboltabelle vollständig war, festgestellt, daß nur 2 Byte Speicherplatz benötigt wurden, so führte dies zu der Fehlermeldung "Phasenfehler" und einem fehlerhaften Maschinencode. Dies wurde in der aktuellen Version beseitigt. Wird ein Phasenfehler erkannt, so wird ein NOP-Befehl eingefügt und im Protokoll durch einen Kommentar gekennzeichnet.

In der vorliegenden Form ist der Cross-Assembler nur auf einer DVA 301 ablauffähig. Soll er auf einer anderen Anlage der 24 bit-Rechnerfamilie ablaufen, so müssen die in PROSA geschriebenen Programmteile überprüft und gegebenenfalls angepaßt werden.

9. Weiterarbeit

Um die Übersetzungszeit noch weiter zu verringern, ist geplant, alle Subroutinen in PROSA umzuschreiben. Die hierdurch und durch den Fortfall der Bibliotheksmoduln, die bei FORTRAN-Programmen benötigt werden, erzielte Speicherplatz-Einsparung dürfte ein lineares Binden ermöglichen. Durch den Fortfall der Plattentransfers beim Segmentwechsel würde zusätzlich eine Verkürzung der Übersetzungszeit erreicht. Eine weitere Verkürzung kann erreicht werden, indem vermieden wird, daß die bereits im 1. Durchlauf erfolgten Assemblertätigkeiten im 2. Durchlauf noch einmal wiederholt werden müssen. Hierzu müßte jede Anweisung nach ihrer Bearbeitung im 1. Durchlauf zusammen mit den ermittelten Parametern in eine Plattenspeicherdatei übernommen werden.

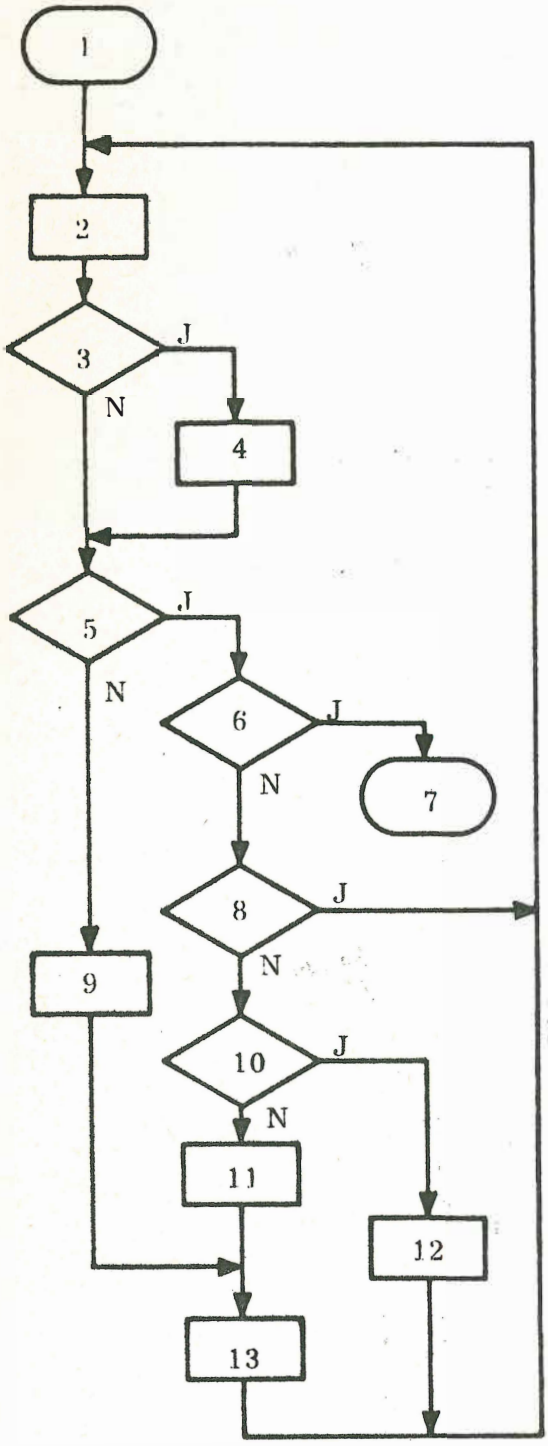
Inzwischen werden in unserem Institut noch die Mikroprozessoren SC/MP der Firma National Semiconductor, MCS 6502 der Firma MOS-Technology und SAB 8080 der Firma Siemens eingesetzt. Die zu diesen Mikroprozessoren gehörenden Cross-Assembler liegen uns bereits als Kartenstapel vor und sollen ebenfalls im Rahmen von Wahl- oder Diplomarbeiten auf der DVA 301 ablauffähig gemacht werden.

10. Quellenhinweise

[1] Tagungsbericht der 7. Jahrestagung (21. bis 23.4.1976) des Siemens Prozeß-rechner-Anwenderkreis SAK1, S.145-152.

[2] Donovan, J.J.: System-Programmierung. Vieweg Verlag, 1976.

[3] M6800 Microprocessor Programming Manual. Fa. MOTOROLA, 1975.



1 Einsprung zum 1. Assemblerdurchlauf!

2 Inhalt einer Karte lesen.

3 Enthält die Anweisung eine Marke?

4 Marke mit dem aktuellen Adreßzählerwert in die Symbol-Tabelle speichern.

5 Ist der Befehl eine Pseudo-Anweisung?

6 Ist die Anwg. eine END-Anweisung?

7 Verzweigen zum 2. Durchlauf!

8 Enthält die Karte Anweisungen, die erst im 2. Durchlauf bearbeitet werden?

9 Befehlscode in Tabelle suchen und Befehlslänge ermitteln.

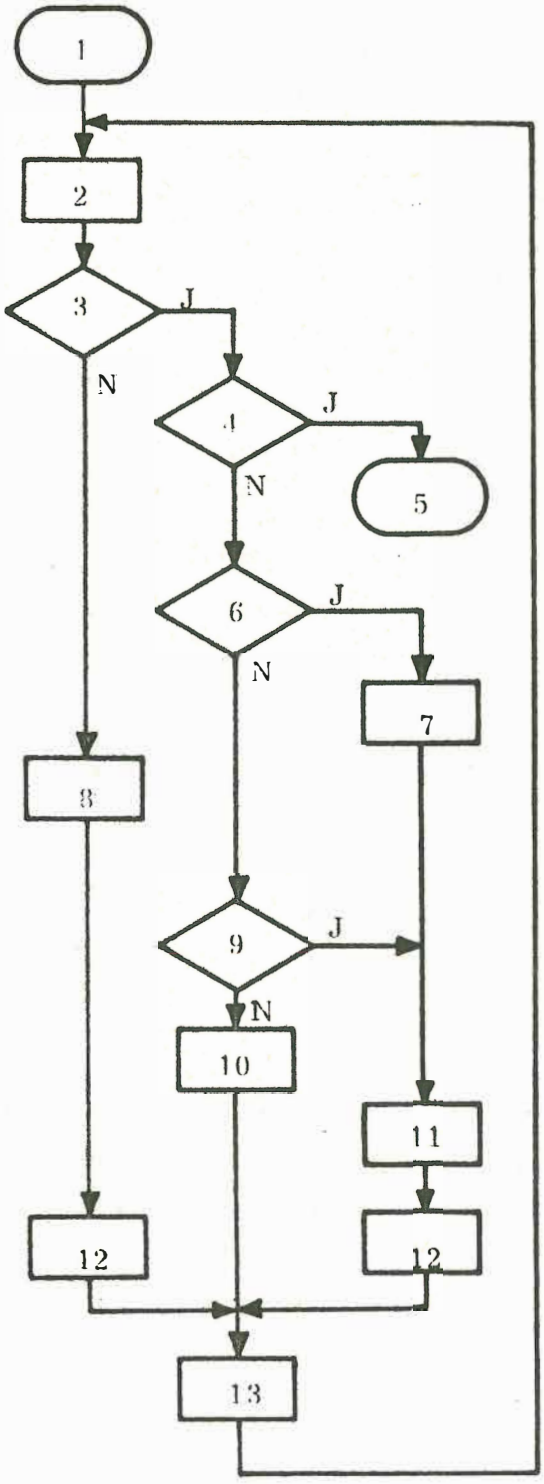
10 Enthält die Karte eine EQU-Anwg.?

11 Erforderlichen Speicherbedarf feststellen.

12 Marke in der Symbol-Tabelle mit dem Wert des Operanden versehen.

13 Adreßzähler verändern.

Bild 1. Ablauf von Pass 1 eines Assemblers (allgemein)



1 Einsprung zum 2. Assemblerdurchlauf!

2 Inhalt einer Karte lesen.

3 Enthält die Karte eine Pseudo-Anweisung?

4 Ist es die END-Anweisung?

5 Ende der Assemblierung!

6 Enthält die Karte eine Anweisung zur Konstanten-Erzeugung?

7 Konstante erzeugen und speichern.

8 Befehlscode in Tabelle suchen, Befehlslänge und Maschinencode bestimmen. Operand auswerten und die Teile des Befehls zusammensetzen und speichern.

9 Enthält die Karte eine Anweisung zur Speicherplatzreservierung?

10 Anweisung ausführen.

11 Anzahl der erforderlichen Speicherplätze ermitteln.

12 Adreßzähler verändern.

13 Anweisungszeile ausdrucken.

Bild 2. Ablauf von Pass 2 eines Assemblers (allgemein)

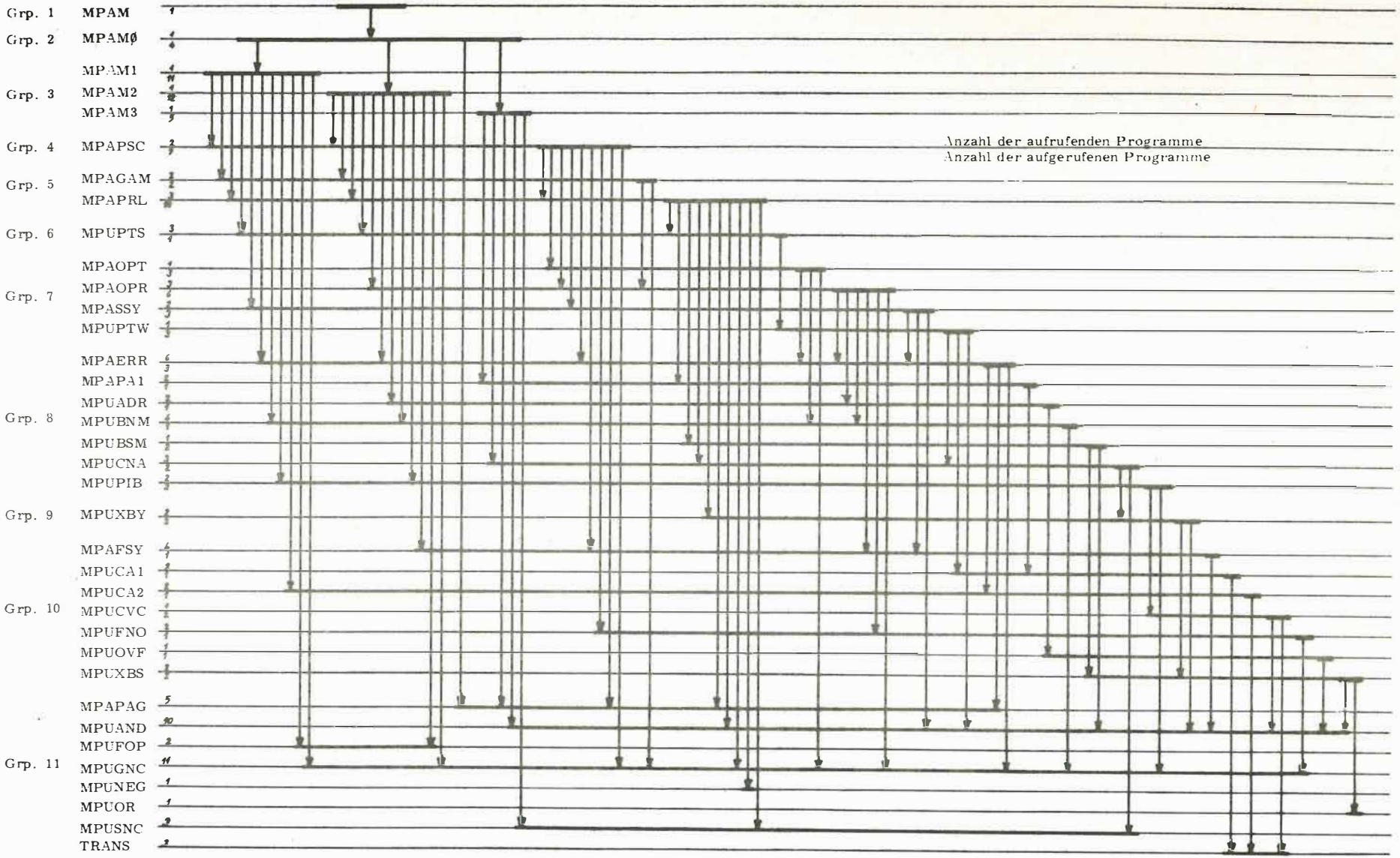


Bild 3. Aufruf-Hierarchie des ursprünglichen Cross-Assemblers

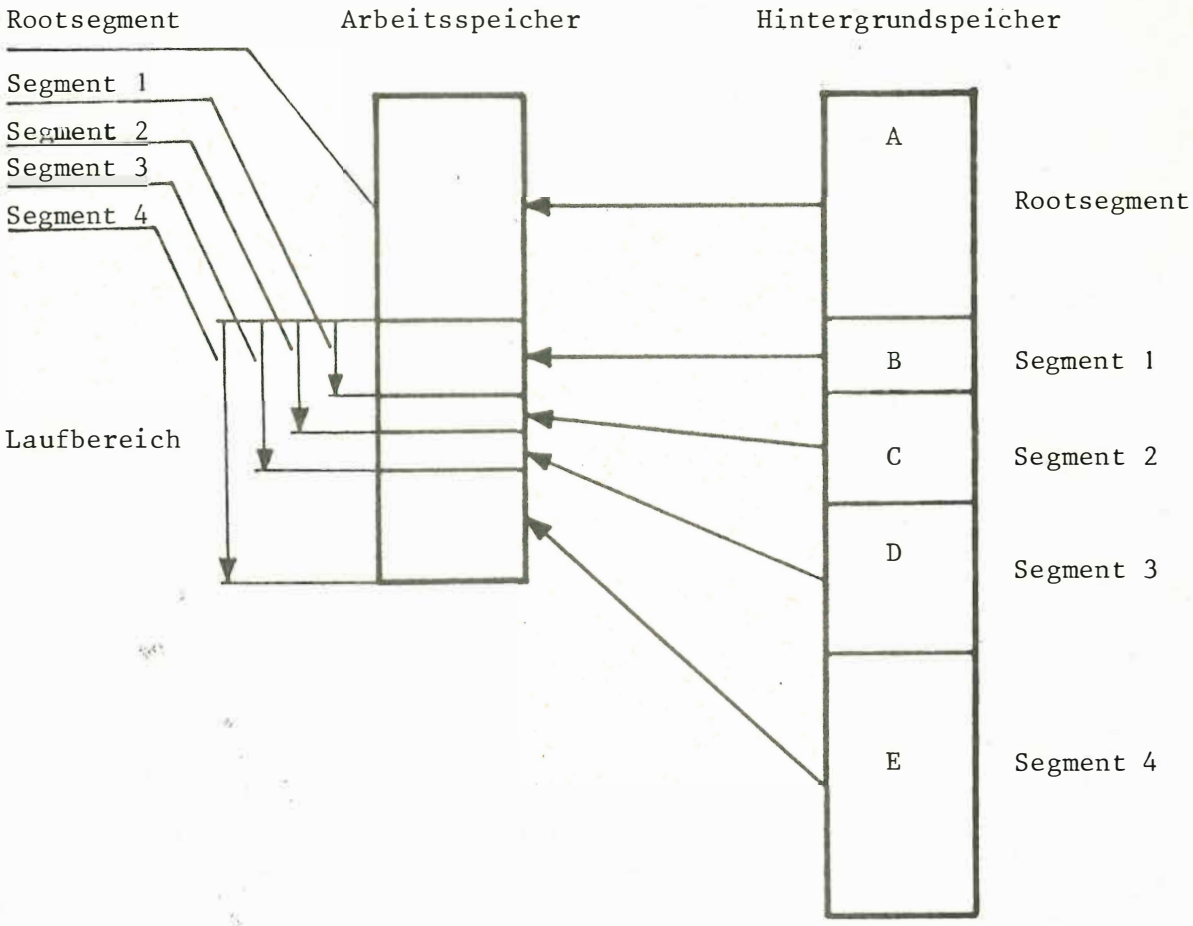


Bild 4. Speicherplatzbelegung bei der einfachsten Overlay-Struktur

Ankopplung eines CDC-Plattenspeichers 564C an eine DVA 301

R. Janssen *

1. Anlagenkonfiguration

Am Institut für Allgemeine Elektrotechnik und Automatisierung ist ein Prozeßrechner-System vom Typ Siemens 301 installiert, dessen Konfiguration im SAK-Tagungsbericht 1976 beschrieben ist. Als Massenspeicher ist ein Siemens-Plattenspeicher PS 21 mit 4,8 Mbit Speicherkapazität angeschlossen. Zur Steuerung dieses Plattenspeichers dienen zwei Geräte:

- Das Grundsteuergerät P3KS 301 im Rechnerschrank
- Das Plattenspeicher-Steuergerät PSPS im Plattenspeicher-Anbauschrank

Das Institut erhielt von der Firma Siemens einen zweiten Plattenspeicher geschenkt, der Kapazität und Schnelligkeit der Rechenanlage erhöhen sollte. Dieser Plattenspeicher vom Typ 546C (Hersteller: Control Data Corporation, CDC) ist jedoch für die Zusammenarbeit mit einem anderen Rechnersystem konzipiert und daher nicht mit dem Siemens-Plattenspeicher kompatibel. Außerdem waren keine Steuergeräte für den CDC-Plattenspeicher vorhanden. Es wurde nach einer Ankoppelmöglichkeit für den CDC-Plattenspeicher gesucht, die ohne Änderung der bestehenden Hardware und Software auskam.

2. Ankoppelmöglichkeiten

Die Entwicklung eines speziellen Steuergerätes für den CDC-Plattenspeicher erschien zu aufwendig, auch bei Verwendung von Mikroprozessoren. Die Verwendung der Siemens-Steuergeräte (PSPS und P3KS 301) setzte voraus, daß ein Anpassungsgerät entwickelt wurde, welches den CDC-Plattenspeicher kompatibel mit dem Siemens-Plattenspeicher macht.Dieses Gerät wird in Abschnitt 3 beschrieben.

---

* Dipl.-Ing. R. Janssen ist Verwalter der Stelle eines wissenschaftlichen Assistenten am Lehrstuhl und Institut für Allgemeine Elektrotechnik und Automatisierung der RWTH-Aachen, Direktor Professor Dr.-Ing. Hans Henning.

Die Ankoppelschaltung für den CD-Plattenspeicher wurde im Rahmen einer dreimonatigen Diplomarbeit vom Verfasser entworfen.

Ein Kauf der beiden Siemens-Steuergeräte schied aus, da die Geräte nicht mehr lieferbar sind. Ein Nachbau dieser Geräte wäre zwar möglich, aber ebenfalls zu aufwendig gewesen. Deswegen wurde ein Umschalter entwickelt, mit dessen Hilfe beide Plattenspeicher die vorhandenen Steuergeräte abwechselnd benutzen können (s. Abschnitt 4). Der Aufwand hierfür beschränkt sich auf rd. 15 integrierte Schaltungen.

3. Anpassungsgerät

Beide Plattenspeicher werden zwar mit den gleichen Signalen gesteuert, sie unterscheiden sich jedoch im mechanischen und elektrischen Aufbau in drei Punkten, die beachtet werden müssen:

- o Plattenanzahl
  Als Datenträger dienen im Siemens-Plattenspeicher zwei rotierende Plattenstapel mit je vier ausnutzbaren Plattenoberflächen. Die beiden Plattenstapel werden übereinander in das Gerät eingesetzt und sind durch die Kodierung der acht Magnetköpfe als zwei getrennte Geräte Eins und Zwei ansprechbar. Der Plattenstapel des CDC-Plattenspeichers hat dagegen zehn ausnutzbare Plattenoberflächen in einem gemeinsamen Plattenstapel. Da das Organisationsprogramm der DVA 301 und die Plattenspeicher-Steuergeräte jedoch nur drei Bits für die Adressierung der Magnetköpfe verwenden, können von den zehn Plattenoberflächen des CDC-Plattenspeichers nur acht benutzt werden. Die beiden mittleren Plattenoberflächen bleiben dadurch frei.

- o Pegel
  Der Siemens-Plattenspeicher arbeitet mit TTL-Pegeln, der CDC-Plattenspeicher mit zwei verschiedenen anderen Pegeln. Das Anpassungsgerät muß daher Pegelumsetzer erhalten, die in Unterlagen der Firma Siemens beschrieben sind. Zusätzlich benötigt der CDC-Plattenspeicher noch vier Versorgungsspannungen, die im Anpassungsgerät bereitgestellt werden müssen.

- o Sektoranzahl
  Die Plattenoberflächen werden bei beiden Plattenspeichern durch die Positionierung der Magnetköpfe in 203 Spuren eingeteilt. Beim Siemens-Plattenspeicher werden die Spuren in 16 gleich lange Sektoren eingeteilt. Um den Sektoranfang erkennen zu können, sind am Umfang der Sektorscheibe, die mit dem Plattenstapel fest verbunden ist, 33 Schlitze gefräst. Sie werden induktiv abgetastet, wobei jeder zweite Schlitz den Anfang eines Sektors kennzeichnet. Der 33. Schlitz (Indexschlitz) kennzeichnet zusätzlich den Sektor

mit der Nummer Null. In einem Sektor werden beim Siemens-Plattenspeicher 64 Datenwörter und 14 weitere Wörter zur Synchronisation und Kontrolle geschrieben. Zwischen den Informationen zweier Sektoren wird eine Lücke gelassen, die bei Nenndrehzahl der Plattenspeicher (2400 Umdrehungen pro Minute) rd. 65 Mikrosekunden lang ist. Sie bewirkt, daß bei Abweichungen der Stapeldrehzahl von der Nenndrehzahl um bis zu 2 % sich die Sektordaten nicht überschneiden können, denn die Zeit für das Schreiben der Sektordaten ist drehzahlunabhängig (quarzgenau).

Der Plattenstapel des CDC-Plattenspeichers ist durch 20 Sektorschlitze und einen Indexschlitz in 20 Sektoren unterteilt. Da die Zeit für das Schreiben eines Bits bei beiden Plattenspeichern 800 ns beträgt, kann in einem der kleineren Sektoren (ein Zwanzigstel der Spur) des CDC-Plattenspeichers nicht die gleiche Datenmenge untergebracht werden wie in einem der größeren Sektoren (ein Sechzehntel der Spur) des Siemens-Plattenspeichers. Man könnte zwar für die CDC-Plattenstapel neue Sektorscheiben anfertigen, jedoch müßte dann auch die Elektronik des Speichers geändert werden, da die Sektorimpulse auch zur Drehzahlmessung benötigt werden. Da diese Lösung aber zu aufwendig ist, werden im Anpassungsgerät die 20 CDC-Sektorimpulse in 32 Impulse für die Siemens-Steuergeräte umgesetzt, so daß für diese Steuergeräte ein Plattenstapel mit nur 16 Sektoren simuliert wird. Die Daten können also auf dem CDC-Plattenstapel in der gleichen Art gespeichert werden wie auf den Siemens-Plattenstapeln. Bei der Umsetzung ist zu beachten, daß die zu erzeugenden 32 Impulse an möglichst vielen Stellen von den ursprünglichen 20 Impulsen abgeleitet werden, damit sich Umsetzungsfehler nicht zu unzulässig großen zeitlichen Abweichungen addieren können. Um dies zu erreichen,werden die 20 Sektorimpulse des CDC-Plattenspeichers während einer Plattenumdrehung um unterschiedliche Beträge (z.T. zweimal) verzögert, so daß sie zeitlich mit den geforderten 32 Impulsen übereinstimmen. Die Verzögerungszeiten werden durch Abzählen der Schwingungen eines 10-MHz-Oszillators erzeugt, dessen Frequenz mit der Plattendrehzahl synchronisiert wird. Dabei ist von Vorteil, daß sich jeder fünfte Sektorimpuls des CDC-Plattenspeichers zeitlich wieder mit jedem achten geforderten Sektorimpuls für die Siemens-Steuergeräte deckt. Dadurch kann sich die Impulsumsetzung viermal während einer Plattenumdrehung in gleicher Weise wiederholen.

4. Umschalter

Der elektronische Umschalter für die beiden Plattenspeicher besteht aus zwei Teilen. Ein Teil (bestehend aus vier integrierten TTL-Schaltungen) kann im Arbeitsspeicher-Nahtstellen-Multiplexer (ASM) des Rechners untergebracht werden.

Er erzeugt aus dem Koppelbefehl BBE des Rechners ein Signal MODULE SELECT für einen der beiden Plattenspeicher. Mit diesem Signal werden in dem so ausgewählten Plattenspeicher im zweiten Teil des Umschalters die ankommenden und abgehenden Signale für das Plattenspeicher-Steuergerät PSPS durchgeschaltet. Das Signal MODULE SELECT für einen Plattenspeicher wird so lange aufrechterhalten, bis der Datenaustausch mit dem Signal BAP vom Plattenspeicher aus beendet wird. Erst dann wird ein eventuell anstehender Koppelbefehl für den anderen Plattenspeicher als Signal MODULE SELECT zu diesem weitergeleitet.

Ausführliche Unterlagen über Anpassungsgerät und Umschalter sind im Institut vorhanden.

Ein 8 K Halbleiterspeichermodul für das Siemens-System 300 - 16 Bit

H. Huppertz, P. Reinhart, W. Tenten

KFA Jülich
Zentrallabor für Elektronik
Nukleare Elektronik

Im Zentrallabor für Elektronik der Kernforschungsanlage Jülich ist seit etwa 2 Jahren ein Rechnerverbund in Betrieb, bei dem unter anderem eine Siemens 306 mit einer Siemens 320 über CAMAC gekoppelt ist.

Die Siemens 320 dient in diesem Verbund als intelligenter Multiplexer und ist Mittelpunkt aller on-line-Datenwege.

Sie übernimmt die Koordination aller Ein- und Ausgabedaten für alle angeschlossenen Rechner und Sichtgeräte. Der Grundausbau der S320 beträgt 8K. [1]

Es hat sich nun nach Inbetriebnahme des Verbundes gezeigt, daß die Ein-Ausgabe-Datenmengen, insbesondere bei Benutzung der angeschlossenen grafischen Sichtgeräte, recht hoch sind und deshalb der 8K-Speicher nicht ausreicht, zumal von diesen 8K auch noch der Programmbereich abzuziehen ist.

Sobald mehrere Benutzer Grafik machen, wird der Bildaufbau stockend und die Wartezeiten werden länger.

Zum anderen erzeugt jede Eingabe in der Regel eine große Anzahl von Ausgabedaten, die sehr schnell von der S306 in die S320 transferiert werden, dann aber nur relativ langsam an die Sichtgeräte und Satellitenrechner ausgegeben werden können. Diese Daten sollten also in der S320 gepuffert werden können.

Außerdem ist es noch wünschenswert, daß jeder Benutzer seinen eigenen Eingabepuffer erhält, damit erst nach Komplettierung der Eingabedaten der Transfer in die S306 zu erfolgen braucht.

Aus diesen Gründen wurde eine weitere Speichermöglichkeit in der S320 erforderlich und damit ein weiterer Ausbau des Zentralspeichers.

Es sprachen nun etliche Punkte für eine Eigenentwicklung eines 8K-Halbleiterspeicherblocks.

1) In der KFA Jülich sind außer der S320 im Zentrallabor für Elektronik noch 3 weitere S330-Anlagen installiert, die alle nicht voll ausgebaut sind, so daß der Speicherblock auch dort Verwendung finden kann.

2) Die Siemens 8K Kernspeichermodule sind Doppelflachbaugruppen und belegen jeweils 2 Steckplätze, wobei jeweils ein Steckplatz nicht benutzt wird. Der Zentralrahmen der 320 kann 32K Speicherkapazität aufnehmen.

   Das 8K Halbleiterspeichermodul besteht aus einer Flachbaugruppe und benötigt im Normalfall nur einen Steckplatz. Da man aber nur jeden zweiten Steckplatz für ein Speichermodul benutzen kann, ist auch hierbei im Normalfall nur ein Ausbau bis 32K im Rahmen Zentraleinheit möglich.

   Flanscht man nun zwei 8K Halbleiterspeichermodule aufeinander, so ergibt sich in der gleichen Bauart wie das Siemens 8K Speichermodul ein 16K Speichermodul, d.h. im Rahmen der Zentraleinheit kann der Vollausbau von 64K erfolgen, allerdings ist es dann notwendig, die höchstwertige Adreßleitung auf einen freien Pin zu verdrahten.

3) Zum Zeitpunkt der Problemstellung waren drei recht preiswerte 4K statische RAMs erhältlich, die in ihrer Zugriffszeit im Bereich der Speicherzugriffszeit der 320 lagen, nämlich etwa 250 ns, so daß die Verwirklichung einfach und preiswerter wurde als ein Kauf von Kernspeichermodulen.

Speicherbausteine

Mit Lieferzeiten von etwa 8 bis 10 Wochen wurden im Februar dieses Jahres

statische 4K x 1Bit RAMs von den Firmen SEMI, AMD und MOSTEK angeboten.

Der Chip der Firma SEMI hat den entscheidenden Nachteil, daß er 3 Versorgungsspannungen braucht. Er ist allerdings der preiswerteste.

Da das Speichermodul nicht ausschließlich für die Anwendung in den Rechnern S320 - S330 benutzt werden soll, sondern auch als Memory in Doppeleuropa und anderen TTL-Systemen, ist es wünschenswert, nur mit einer Versorgungsspannung von +5V zu arbeiten.

Für das Speichermodul waren also nur der Baustein von AMD und MOSTEK interessant. Diese beiden RAMs sind sich sehr ähnlich.

Die Zugriffszeit liegt bei etwa 200 ns und die Zykluszeit bei 350 ns.

Das Timing beider Bausteine ist etwa gleich, beide sind TTL-kompaktibel und haben ein Fan-out von 2 TTL-Lasten.

Des weiteren gestatten beide Bausteine einen Read-Modify-Write-Zyklus, und diese Betriebsart ist besonders gut geeignet, um das 4K-RAM an das vorhandene Timing der S320 anzupassen.

Allerdings unterscheiden sich die beiden Bausteine auch in einigen interessanten Punkten.

Zum Beispiel hat der AMD-Chip ein 22 Pin-Gehäuse und der MOSTEK-Chip ein 18 Pin-Gehäuse.

Die Leistungsaufnahme des AMD-RAMs ist etwa dreimal so hoch wie die des MOSTEK-RAMs, nämlich 580 mW. Das MOSTEK-RAM benötigt maximal 220 mW.

Die Ruheverlustleistung beträgt ca. 100 mW beim AMD-RAM und 50 mW beim MOSTEK-RAM.

Wegen der letztgenannten Eigenschaften wurde das MOSTEK-RAM dem AMD-RAM vorgezogen.

Realisierung des 8K-Speichermoduls

Der 8K-Halbleiterspeicher ist eine Flachbaugruppe, die den Flachbaugruppen der S320 entspricht, und zwar in Maßen und Steckern. Die Flachbaugruppe ist bestückt mit 36 statischen 4K x 1 Bit-RAMs von MOSTEK Typ MK 4104P.

Die Wortlänge des Speichers beträgt 18Bit, bestehend aus 2 Byte mit je einem Paritybit, so daß auch Byte-Operationen möglich sind.

Bei der S320 wird das Paritybit allerdings nicht genutzt, da nur die DMA-Schnittstelle ein Paritybit erzeugt, die S320 aber keine Parityprüfung macht.

Das Speichermodul entspricht also in der Organisation dem Mini-Store 333 und ist somit auch in der S330 verwendbar.

Die Stromaufnahme beträgt im Betrieb pro Flachbaugruppe ca. 0,5 A, d.h. bei Vollausbau mit 64K Speicher 3,5 A.

Dieser Strom wird vom 50A-Netzteil der S320 aufgebracht, zudem ist noch genügend Reserve vorhanden, da die Zentraleinheit mit 8K Kernspeicher 15A zieht.

Insgesamt sind also mit 64K Speicher 18,5A aufzubringen.

Die Flachbaugruppen in der S320 sind etwas kleiner als das Doppeleuropa-format. Es ist also möglich, mit einem anderen Plattenzuschnitt den Halbleiterspeicher auch als Doppeleuropakarte zu fertigen.

Zur Betriebserfahrung ist zu sagen, daß der Speicher seit 2 Wochen im 24-Stunden-on-line-Betrieb ohne Störung läuft. Längere Erfahrung liegt leider noch nicht vor. Im Test wurde der Speicher mit Bitmustern, random-Daten und random-Adressen betrieben. Außerdem wurde über längere Zeit ein Temperatur-test mit 40°C Umgebungstemperatur gemacht und kurzzeitig, das heißt im Stundenbereich, mit einer Temperatur von 50°C.

---

[1] Kopplung von verschiedenen Experimentrechnern mit einer S306-320 Installation F. Rongen

Betr.: Vortrag SAK-Tagung 21. 4. 1977 in Dortmund
Thema: Die Fa. Dr. Hell und der Einsatz von Siemens-Prozeßrechnern

Meine Damen und Herren!

Ich möchte Ihnen an dieser Stelle eine kurze Übersicht über die Firma Dr. Ing. Rudolf Hell geben. Zunächst möchte ich mich bedanken beim SAK-Vorstand für die Einladung zu diesem Vortrag.

Da in der Firma Dr. Hell seit einiger Zeit Prozeßrechner 3ØØ-16 Bit eingesetzt werden, ist es sicherlich für die SAK-Mitglieder interessant zu erfahren, welche Aufgaben die Fa. Dr. Hell über diese Rechner abwickelt.
Einigen Zuhörern wird die Fa. Dr. Hell bereits bekannt sein, denn es gibt eine Vielzahl von Hellgeräten in den verschiedensten Industriebereichen.
Seit 1945 hat die Firma ihren Sitz in Kiel. Hier arbeiten ca. 1.5ØØ Mitarbeiter in 3 Werken. Die Firma wurde 1929 in Berlin von Dr. Hell gegründet. Die erste Erfindung war der weltbekannte Hell-Schreiber.
Im Jahre 1971 wurde die Firma in eine GmbH umgewandelt. Siemens ist an dem Stammkapital zu 80% beteiligt. Die Fa. Dr. Hell wird als Siemens-Tochtergesellschaft geführt. Das Stammkapital beträgt 12 Millionen DM. Der Firmengründer, Herr Dr. Hell, ist heute 75 Jahre alt und leitet die Geschicke der Firma als Aufsichtsratsvorsitzender.
Der Jahresumsatz der Firma beläuft sich auf ca. 12Ø Mio-DM.
Der Umsatz ist stark exportorientiert. Ca. 73% des Umsatzes liegen im Exportgeschäft. Davon werden allein 15% über eine Hellgesellschaft in den USA umgesetzt. Neben dem westeuropäischen Markt ist das Japangeschäft und das Ostblockgeschäft bedeutend. Selbst Rot-China wird mit Hell-Geräten beliefert.
Ca. 57% des Umsatzes entfallen auf die Reproduktionstechnik (hauptsächlich CHROMAGRAPH DC3ØØ, ein elektronisches Farbauszugsgerät und KLISCHOGRAPH, eine Graviermaschine)
Ca. 25% des Umsatzes liegen im Bereich der Informationstechnik (Telebild, Faksimilegeräte, Morsegeräte)
Ca. 14% des Umsatzes erstreckt sich auf den Bereich Satztechnik (Lichtsatzgeräte DIGISET)

Der Rest ergibt sich aus Geräten zur Microverfilmung (DICOM) und Geräten für die Textilindustrie.
Die Firma Hell gibt jährlich ca. 11% des Umsatzes für die Forschung und Entwicklung aus (rd. 14 Mio-DM). Wofür ein Teil dieser Mittel eingesetzt wird im Bereich der Prozeßrechnerprogrammierung sei nun näher erwähnt.
Die Firma Dr. Hell wird heute in den drei Bereichen Entwicklung, und Fertigung, Vertrieb und kaufm. Abteilungen von drei Geschäftsführern geleitet. In den Bereich Entwicklung fällt die Hardware-Entwicklung in verschiedenen Laborbereichen und u.a. die Software-Entwicklung. Damit komme ich kurz zu meiner Person: ich leite die Software-Entwicklung im Hause Hell für die anschließend näher beschriebenen Aufgaben. Im Rahmen der Software-Entwicklung sind z.Zt. 16 Programmierer eingesetzt. Die zur Zeit anliegenden Aufgaben er-

fordern die Einschaltung eines zusätzlichen Programmierbüros, das sich mit 6 Programmierern an den Aufgaben der Fa. Hell beteiligt, so daß z.Zt. 22 Programmierer Software-Anwenderpakete für unsere Aufgaben erstellen.
Wie Sie aus der Übersicht sahen, begann die Fa. Hell im Jahre 65 mit der Entwicklung der DIGISET-Anlage, einer schnellen Lichtsetzanlage. Bis dahin hatte die graphische Industrie einen Dornröschenschlaf gehalten. Jahrhunderte blieb das Gebiet der "Satztechnik", von einigen Neuerungen auf dem Gebiet des Bleisatzes abgesehen, auf demselben Entwicklungsstand stehen.
Es gab neben dem Handsatz Maschinen für den Einzelbuchstabensatz oder Zeilensatz. Technische Fortschritte gab es nur dadurch, daß das Tasten der Texte von der Setzmaschine auf Perforatoren mit Zählwerken übertragen wurden. An diesen Perforatoren konnte dann ein Setzer einen Text gestalten und der dabei gewonnene Lochstreifen steuerte dann die Bleisetzmaschine. Die Arbeitsplätze der Setzer wurden damit moderner, doch die Arbeit des Aufteilens der Zwischenräume einer Zeile, die Durchführung von Silbentrennungen usw. blieb beim Setzer selbst. Im Jahre 65 gab es dann in Deutschland die ersten Versuche (erfolgreiche Versuche), diese Arbeit zu rationalisieren. Man schrieb einen Text an einem Perforator ohne die Setzmaschinencodes und ohne Berücksichtigung des Zeilenfalls einfach "endlos". Dieser gewonnene Endloslochstreifen wurde dann an einen"Satzrechner" gegeben, der die Berechnung der Zeilen und auch die Silbentrennung durchführte. Das Ergenis war eine Ausgabe eines Steuerlochstreifens für die Setzmaschine.
Obwohl Hell damals bereits die Lichtsetzanlage DIGISET entwickelte, die solche Bleisetzmaschinen ersetzen bzw. ablösen sollte, war die Firma aus der Marktsituation heraus gezwungen, auch Programme für Bleisetzmaschinen zu entwickeln. Damals wurde der Rechner 3Ø3, (später 3Ø2 und 3Ø4) als Bleisatzrechner eingesetzt. Hell führte im Jahre 65 in Deutschland gleichzeitig mit einem IBM-Einsatz die die ersten Rechnerprogramme in der graph. Industrie ein. Zur Steuerung der Lichtsetzanlage Digiset waren selbstverständlich auch Programme erforderlich, die die typografischen Erfordernisse der verschiedenen Satzaufgaben abdecken mußten. Entsprechende Programme wurden hier von Hell für die DVA 3ØØ3 und die Prozeßrechner 3Ø3 (3Ø2 und 3Ø4) entwickelt.
In Zusammenarbeit mit der Fa. Siemens wurden dann Programmsysteme für die 4ØØ4-Rechner entwickelt. Hier übernahm dann die 4ØØ4 die Aufbereitung, Verwaltung und Korrektur sowie die Ausgabe der Texte an die Lichtsetzanlage (Beispiele f. Satzgestaltung glatter Text und Tabellensatz als Filmfolie)
Diese Satzprogramme werden laufend ergänzt um wichtige Funktionen der Setzereien, wie z.B. Redaktionssystem usw.
Um zu den kleineren Systemen der Konkurenz beim Kunden bestehen zu können, hat die Fa. Hell im Jahre 72 ein eigenes kleineres Satzsystem erstellt. Das geschah zunächst für die Siemensanlage 4Ø4. Nach Auslaufen dieses Modells hat sich die Fa. Hell entschlossen, als Satzrechner die Prozeßrechner der 3ØØ-16-Bit-Serie einzusetzen. Heute läuft das komplette Satzsystem eines Setzereibetriebes über einen Prozeßrechner 33Ø ab, der in typischer Anlagenausstattung einen Kernspeicherausbau von 4Ø bis 64 KW hat, Lochstreifenein abegeräte, Magnetbandgerät, Datensichtgeräte zur Texterfassung und Korrektur, Plattenspeicher zur Textspeicherung (2 Laufwerke zu je 6Ø MB-Speicherkapazität) und der Lichsatzanlage DIGISET (zur Ausgabe der aufbereiteten Texte oder Seiten). Für diese typische

Anlagenkonfiguration wurde ein komfortables Satzprogramm erstellt, das es dem Anwender ermöglicht, sowohl Zeitungs-, als auch Werksatz zu produzieren. Diese Systeme werden in naher Zukunft ergänzt um Umbruchsysteme für den redaktionellen Umbruch, Anzeigenumbruch und Werksatzumbruch. Bei diesen kompletten Systemen übernimmt die DVA 33Ø nicht nur die Aufbereitung der eingegebenen Texteinheiten, sondern sie übernimmt auch wesentliche Teile der Steuerung der Digisetaufzeichnungseinheit. Diese Steuerung wurde bei den früheren Digisetmodellen hardwaremäßig vorgenommen. Aus Kostengründen und Gründen der schnelleren Reaktion auf Kundenanforderungen wird diese Steuerung jetzt von einem Prozeßrechner vorgenommen. In obigem Fall also eine DVA 33Ø und in den Fällen, wenn die Satzaufbereitung durch Fremdrechner erfolgt, entweder durch die DVA 31Ø oder DVA 33Ø mit einem speziellen Steuerprogramm. Um Ihnen eine Größenordnung der bisher eingesetzten Systeme zu geben, hier folgende Zahlen: bis Jahresende werden an ca. 14 Kunden 18 Großsysteme mit obiger Peripherie ausgeliefert sein (Unterschied der Zahlen ergibt sich daraus, daß einige Anwender ein back-up-System haben). Reine Aufzeichnungseinheiten mit Steuerrechner 31Ø werden dann ca. 16 ausgeliefert sein. An Großsystemen wird mit einer jährlichen Steigerungsrate von 10-12 Systemen gerechnet und an Steuereinheiten mit 31Ø ca. 20 Anlagen.
Doch nicht nur das Gebiet der Satztechnik wird von uns mit dem Siemensrechner 31Ø ausgestattet, sondern diese Anlage findet bei uns im Hause auch Anwendung auf dem Gebiet der Mikroverfilmungsanlagen. Die Firma Dr. Hell baut die DICOM-Anlage zur Mikroverfilmung in den Maßstäben 1:24, 1:42, und 1:48.
Damit kann Rollfilm oder Mikrofiche erstellt werden. Ausgehend von Magnetbanddateien der Anwender übernimmt hier ein entsprechendes Programm der DICOM-Anlage das Setzen und Gestalten der Mikrofiches. Unter Gestaltung ist hier zu verstehen, daß nach den Angaben des Anwenders aus den druckaufbereiteten Magnetbändern Suchbegriffe ermittelt werden, die auf dem Mikrofiche in einer Indexseite oder lesbar im Kopf (als eye-ball-Zeichen) dargestellt werden. Auch hier wird mit größeren Stückzahlen für Kundeneinsätze gerechnet (Banken, Sparkassen, Großbetriebe, Serviceunternehmen zur Mikroverfilmung).

Zum Schluß noch weitere Anwendungsgebiete, in denen die Fa. Hell in Zukunft Prozeßrechner einsetzt bzw. eingesetzt hat. Hier ist das Gebiet der Textilindustrie zu nennen, wobei hier heute ein 31Ø-System mit einer Floppy-disk innerhalb des Hell-Patro-Systems arbeitet. Es ist hier möglich, die in der Textilindustrie vorkommenden Dessins für den Textildruck sowie Skizzen und Patronen für die Webereien im Rechnersystem zu speichern, zu korrigieren über einen Farbmonitor und die gespeicherte Information zu Jacquardinformation für die Webereien auszugeben. Ferner sollen Farbauszüge für den Textildruck zur Gravur von Kupferwalzen für den Druck von Transferpapieren über diese Systeme verarbeitet werden. An diesen Systemen wird zur Zeit gearbeitet. Erste Anlagen der Patro-Familie sind bereits bei Webereien eingesetzt.
Die Entwicklung im Hause Hell wird dahin gehen, daß bei neuen Scannersystemen auch Prozeßrechner eingesetzt werden können. Die durch die Scanner erfaßten Daten für Katalogseiten, Magazinseiten usw. werden auf Großplatten verwaltet und zur Gravur der Zylinder aufbereitet. Über diese umfangreichen Systeme sollte, falls Interesse besteht, getrennt berichtet werden.
In Anbetracht der fortgeschrittenen Zeit hoffe ich, daß Sie einen Überblick über die Fa. Hell bekommen haben und die verschiedensten Anwendungsgebiete, in denen wir Prozeßrechner heute einsetzen.
Ich danke für Ihre Aufmerksamkeit.

Dortmund, 22. April 1977
Ca/Go

Rechner-System 300 steuert und regelt den Verkehr

Camen Siemens ZN DO

1. Einleitung

Eine zügige Abwicklung des Straßenverkehrs ist heute ohne Straßenverkehrssignalanlagen (SVA) nicht mehr denkbar.
Die Anlagen müssen folgenden gesetzlichen Vorschriften entsprechen:
Straßenverkehrsordnung (StVO Ø 37)
Richtlinien für Entwurf, Bau und Betrieb von Lichtsignalanlagen im Straßenverkehr (RiL SA)
VDE-Bestimmungen für Straßenverkehrs-Signalanlagen (SVA) 0832/4.75, DIN 57 832
Sie haben die Aufgabe, an einer Einmündung oder Kreuzung

1. Die Sicherheit zu erhöhen, d.h. Gefährdungspunkte zu vermeiden
2. Die Leistungsfähigkeit zu erhöhen, d.h. den Verkehr flüssig zu halten und die größtmögliche Verkehrsmenge über die Kreuzung zu bringen
3. Die Verkehrsströme für den Verkehrsteilnehmer übersichtlich zu lenken und
4. für einen wirtschaftlichen Verkehrsablauf zu sorgen

Dies ist vor allem für die Innenstadtbereiche sehr bedeutungsvoll, weil dort die Schaffung breiterer oder neuer kreuzungsfreier Verkehrswege große Finanzmittel in Anspruch nimmt, die nicht immer zur Verfügung stehen.
Die großräumige Verkehrslenkung verlangt daher den Einsatz moderner Steuerungsmittel, wie z. B. das Rechner-System 300, das auch als Verkehrsrechner (VSR) eingesetzt wird.

2. Planung

Für die Steuerung der Straßenverkehrssignalanlagen sind Signalzeitenpläne erforderlich. Die Faktoren für die Berechnung dieser Signalzeitenpläne sind

1. Die Verkehrsmenge, die durch eine Verkehrszählung ermittelt wird
2. Die Verkehrszusammensetzung, Anteil der Zweiräder, PKW, LKW und LKW mit Anhänger
3. Die zulässige Geschwindigkeit, maximal 70 km/h
4. Die Unfallhäufigkeit (Untersuchung über die Unfallursachen)
5. Die Sichtverhältnisse

Aus der ermittelten Verkehrsmenge wird der Strombelastungsplan erstellt, der einen Überblick über die einzelnen Verkehrsströme gibt. Die zueinander verträglichen Verkehrsströme werden zu Phasen zusammengefaßt. Die Stärke der abbiegenden Ströme ist entscheidend, ob die Kreuzung zwei- oder mehrphasig geregelt werden muß.

Die einzelnen Phasen werden durch eine Zwischenzeit, bestehend aus der Räumzeit, die durch die Geometrie der Kreuzung bestimmt wird, und den Übergangszeiten, 3 - 5 Sekunden gelb (3 S bei 50 km/h, 4 S bei 60 km/h, 5 S bei 70 km/h) und 2 - 3 Sekunden rot-gelb, voneinander getrennt.

In diese Struktur werden die Grünzeiten eingefügt, für Fahrzeuge mindestens 7 Sekunden und für Fußgänger mindestens 5 Sekunden. Die Länge der Grünzeit ergibt sich aus der Verkehrsmenge, die in einer Phase die Kreuzung passieren soll oder kann. Die Summe der Grünphasen und Zwischenzeiten ergibt im Normalfall Umlaufzeiten zwischen 45 und 90 Sekunden. Längere Umlaufzeiten bedeuten längere Wartezeiten, jedoch auch eine größere Verkehrsleistung.

Ob der Knotenpunkt mit mehr als einem Signalzeitenplan geregelt werden muß, hängt vom Tagespegel ab. Bei stark schwankendem Tagespegel bieten sich zwei Regelverfahren an:

1. Die Festzeitsteuerung mit mehreren Signalzeitenplänen, die über eine Wochenzeitautomatik geschaltet werden.
2. Die verkehrsabhängige Signalzeitenplanauswahl oder Modifikation der Grünzeit in einer vorgegebenen Struktur. Bei diesem Verfahren muß die Verkehrsbelastung über Induktionsschleifen, die in der Fahrbahn verlegt sind, ermittelt werden. Es gibt Zähl- und Anwesenheitsdetektoren mit Betriebsfrequenzen zwischen 44,8 und 62,6 kHz.

3. Steuerungsverfahren

Um ein leistungsfähiges Verkehrsnetz zu erhalten, müssen die einzelnen Straßenverkehrssignalanlagen in den Straßenzügen und Stadtbereichen miteinander koordiniert werden. Aus der Praxis haben sich zwei Steuerungsverfahren entwickelt, die heute eine weite Verbreitung finden:

1. Die Einsatzpunktsteuerung
   Die Signalzeitenpläne werden im Steuergerät an der Kreuzung gespeichert; die Fortschaltung der Signalzeitenpläne, der Signalzeitenplanwechsel und die Synchronisierung erfolgt von der Verkehrszentrale.
2. Die SignalgruppenFernsteuerung
   Die Signalzeitenpläne werden in der Verkehrszentrale gespeichert. Das Schaltgerät an der Kreuzung setzt nur die Befehle um und schaltet die Signale. Der Vorteil dieses Verfahrens ist die absolute Freizügigkeit der Signalprogrammgestaltung durch die Verkehrszentrale.

4. Verkehrsrechner

Aufbauend auf den langen Erfahrungen mit der Zentralsteuerung durch Relaiszentralen wurde der Verkehrsrechner VSR 16 000 entwickelt und im April 1965 in Berlin zum ersten Mal eingesetzt.

Erst der Verkehrsrechner schaffte die Voraussetzung für eine größere Flexibilität der Verkehrszentrale , denn nur mit ihm ist es möglich, ein Straßennetz verkehrsabhängig zu steuern, d. h. die Signalzeitenpläne dem Verkehrs anzupassen.

Die Grundlage dafür ist die Messung bestimmter Verkehrskriterien an den Kreuzungen und die Übertragung der Informationen zum Rechner. Dieser ermöglicht die Meßwertverarbeitung im Echtzeitbetrieb. So können aus vielen Verkehrsmeßdaten binnen einer Sekunde Ergebnisse in Form von Befehlen abgeleitet werden, die dem aktuellen Verkehrsgeschehen entsprechen.

Durch die Verwendung freiprogrammierbarer Speicher können sowohl Programme als auch Daten schnell und einfach über Blattschreiber oder Lochstreifen geändert bzw. ausgetauscht werden.

Die Protokollierung der Betriebsdaten erfolgt mit Datum und Uhrzeit über dem Blattschreiber oder auf Magnetbänder.

5. Verkehrsrechnersystem VSR 16 000

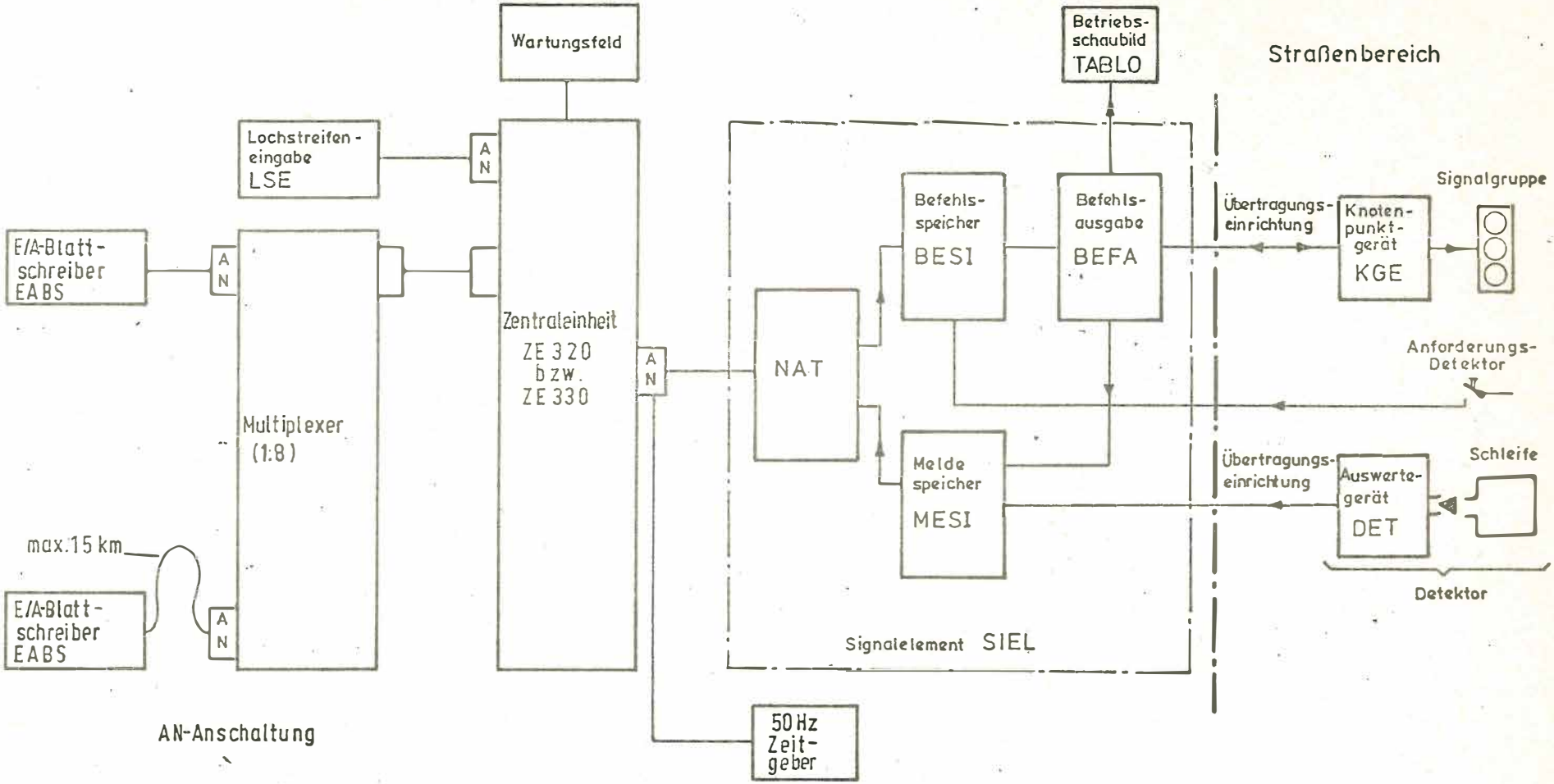
Die Zentraleinheit (ZE) des Verkehrsrechners VSR 16 000 bildet das Rechnersystem 300. Dieses System umfaßt eine auf verschiedene Anforderungen abgestimmte Familie einander verwandter Rechner. Aus der Systemgruppe mit einer Wortlänge von 24-bit wurden die Typen 302, 303 und 304 und aus der Systemgruppe mit einer Wortlänge von 16-bit die Typen 320 und 330 als Verkehrsrechner eingesetzt.

Die einzelnen Systemgruppen haben alle den gleichen Aufbau des Befehls- und Datenwortes und dieselbe Organisation. Die Systemgruppe 24-bit wird einheitlich in der maschinenorientierten Programmiersprache PROSA 300 und die Systemgruppe 16-bit in Assembler programmiert.

Alle Modelle haben genormte "Standard-Nahtstellen" zum Anschluß verschiedener "externer Elemente", wie Blattschreiber, Lochstreifengeräte, Speicher, Datensichtstationen u.s.w.

6. Die technische Struktur

Am Beispiel des Verkehrsrechners VSR 16 030 soll die techn. Struktur erklärt werden. Der VSR 16 030 besteht aus der Zentraleinheit (ZE) mit dem Steuerrechner (STRE) 330 und dem Signalelement (SIEL). Das Signalelement besteht aus der Nahtstellensteuerung (NAT), dem Meldespeicher (MESI), dem Befehlsspeicher (BESI) und der Befehlsausgabe (BEFA).



Tag	Ausgabe	Mitteilung	Name

Blockschaltbild des Verkehrsrechner VSR 16030

SIEMENS AG

Blatt

Blätter

6.1. Die Nahtstellen-Steuerung (NAT)
Die Nahtstellen-Steuerung belegt in der E/A-Anschlußstelle des Steuerrechners einen Anschluß. Sie steuert den Datenfluß vom Steuerrechner zum Befehlsspeicher (Befehle zu den Knotenpunktgeräten) und vom Meldespeicher zum Steuerrechner (Detektormeldungen und Betriebsmeldungen von den Knotenpunktgeräten). Ein Zeitgeber (ZG) erzeugt aus der Netzfrequenz bzw. der Wechselrichter-Frequenz alle 20 ms einen Taktimpuls, der im Steuerrechner bestimmte Programmabläufe auslöst.

6.2. Der Meldespeicher (MESI)
Dem Meldespeicher fällt die Aufgabe zu, die von den Detektoren direkt erfaßten Verkehrsdaten, das sind in erster Linie Anwesenheits- und Anforderungsmeldungen, zwischenzuspeichern und für die Abfrage durch den Steuerrechner bereitzustellen. Entsprechend den Meßaufgaben stehen zählende, speichernde oder nichtspeichernde Eingangsbaugruppen zur Verfügung.

Die Abfrage wird vom Programm bestimmt und richtet sich nach der Meßaufgabe. So wird z. B. zur Ermittlung der Verkehrsstärke durch Zählung alle 20 ms abgefragt und zu Minutenwerten aufaddiert und als 1/4 h-Wert ausgedruckt, für die Erfassung der Anwesenheit eines Fahrzeuges in Intervalle von 20 ms aufaddiert zu 1 Sekunde.
Mit dem 20 ms Abfragetakt ist der Meßwert digitalisiert, und über das Programm kann der Rechner Fahrzeugzahl, Fahrzeugart, bzw. Geschwindigkeit in der betreffenden Spur ermitteln.

6.3. Der Befehlsspeicher (BESI)
Der Befehlsspeicher übernimmt die vom Steuerrechner ausgegebenen Adressen- und Informationsimpulse (0 oder 1), decodiert sie und leitet sie an seine Speicherelemente weiter. Durch diese werden die eigentlichen Befehle für die Signalgruppenfernsteuerung der Kreuzungen (Ende Rot und Ende Grün) gebildet und zur parallelen Übertragung an die

Schaltgeräte an den Knotenpunkten bereitgestellt.

Bei der Einsatzpunktsteuerung werden die Signalisierungszustände schrittweise durch Fortschaltimpulse weitergeschaltet, die nacheinander auf der Leitung übertragen werden.

6.4. Die Befehlsausgabe (BEFA)

Die Befehlsausgabe setzt die vom Befehlsspeicher abgegebenen Signale (0 und 1) auf die Übertragungsleitungen zu den Knotenpunktgeräten um. Außerdem nimmt sie von dort Rückmeldungen über den Betriebszustand (Störungsmeldungen, Ortsprogrammbetrieb u.s.w.) entgegen und sorgt für deren Weiterleitung an den Meldespeicher und - bei Bedarf - an ein Betriebsschaubild.

Um der Forderung nach Einsparung von Leitungen entgegenzukommen, steht eine Befehlsausgabe zur Verfügung, die ein Tonfrequenz-Multiplexsystem (TST 20) enthält (Frequenzband 350 - 3315 Hz, Bandbreite 140 Hz), das es gestattet, einen Knotenpunkt über ein oder zwei Adernpaare (pro DA 14 Kanäle) zu steuern.

7. Die Stromversorgung

Die Betriebszuverlässigkeit der Verkehrssteuerung hängt mit von der steten Bereitschaft der zentralen Stromversorgung ab. Diese muß deshalb so ausgeführt sein, daß ein Höchstmaß an Sicherheit gegeben ist.
Bei den Verkehrsrechnern werden als Grundspannung für die Versorgung des Rechners und der Fernsteuerung einheitlich 60 V verwendet. Diese Spannung wird in einem gemeinsamen Stromversorgungsgerät erzeugt und stabil gehalten. Bei Netzausfall wird automatisch auf Batterie umgeschaltet. Dies erfolgt unterbrechungsfrei und ohne Auswirkung auf den Programmablauf.

Für die mit Wechselstrom gespeisten Lütterbaugruppen der Zentraleinheit ist ein bei Netzausfall anlaufender Gleichstrom-Wechselstrom-Umformer vorgesehen. Werden Trommel- oder Plattenspeicher verwendet, wird ein kontaktlos arbeitender 3-Phasen-Wechselrichter eingebaut, der den höheren Leistungsbedarf deckt und die Frequenz sehr genau konstant hält.

8. Konstruktiver Aufbau

Die Grundeinheiten bestehen aus mehrzeiligen SIVAREP-A- oder -B-Rahmen, die die steckbaren Flachbaugruppen aufnehmen. Die Rahmen werden in Schränke H 1800, B 700, T 700 mm eingebaut. (Schranksystem 8 MF nach DIN 41488)
Die Schränke sind mit Ausnahme des Verteilerschrankes mit einer festen und einer schwenkbaren Einbauebene versehen.

Der Schrank für die Zentraleinheit enthält in der Tür einen Ausschnitt für das Wartungsfeld mit versperrbarer, dichter Klarsichtabdeckung.

9. Programmierte Struktur

Es kristallisiert sich immer mehr heraus, daß gerade die programmierte Struktur die Leistungsmerkmale eines Systems entscheidend beeinflußt. Für die Verkehrsrechner wurde ein neues Konzept entwickelt, das die Möglichkeit bietet, direkt vom Programm Steuerfunktionen auf die Peripherie und damit auf die Knotenpunktgeräte auszuüben. Da die Programme leicht abzuändern sind und damit schnell wechselnden Erfordernissen angepaßt werden können, ergibt sich eine große Flexibilität des gesamten Systems.
Diese Programme liegen in Form von sogenannten Steuerprogrammen vor, die folgenden Aufbau haben und in drei größere Komplexe unterteilt werden:

1. dem Organisationsprogramm, das die Aufgabe hat, den Dialog zwischen dem Bedienungspersonal und dem Rechner über einen Bedienungsblattschreiber zu steuern sowie die formalen und organisatorischen Arbeiten bei Datenverkehr zwischen mehreren externen Elementen, wie die Trommelspeicher oder Plattenspeicher, Lochstreifenelement u.s.w. abzuwickeln und zu koordinieren (Simultanbetrieb).
2. den Programmen für die verschiedenen Regelungsebenen, wie Festzeitsteuerung, Signalprogrammauswahl, Signalprogrammodifikation und den für diese Verfahren entsprechenden Meßwerterfassungs- und Statistikprogrammen.
3. dem Schaltprogramm für Ein-, Ausschalten und Wechseln von Signalzeitenplänen mit integrierten Bearbeitungsroutinen für Signalgruppen- und Einsatzpunktsteuerung

Das Steuerprogramm setzt sich aus sogenannten Programmsätzen (Bausteinen) zusammen, die hinsichtlich ihrer Struktur und Funktion programmtechnisch eine abgeschlossene Einheit bilden.

Für die 16-bit-Rechner wie auch für die 24-bit-Rechner steht eine umfangreiche Programmbibliothek zur Verfügung, aus der ein individuelles Steuerprogramm zusammengestellt werden kann.

10. Einsatzstufen der Verkehrsrechner

Die derzeitigen Verfahren zur Steuerung des Verkehrsablaufes innerhalb von koordinierten Netzen mit Hilfe von Verkehrsrechnern liegen zwischen der Festzeitsteuerung und der Signalprogrammodifizierung. Reduziert man die möglichen Verfahren, die in komplizierten Netzen zur Anwendung kommen könnten, auf ihren wirtschaftlichen Wert, so kommen im wesentlichen drei Steuerungsarten zur Anwendung:

1. die Festzeitsteuerung
2. die verkehrsabhängige Signalprogrammauswahl und
3. die verkehrsabhängige Signalprogrammodifizierung

Diese Stufen bieten dem Verkehrsingenieur die Möglichkeit, den Rechnereinsatz - angefangen von der Festzeitsteuerung bis zur kompliziertesten Verkehrsabhängigkeit - in harmonischer Folge zu planen.

10.1. Festzeitsteuerung

Die erste Einsatzstufe wird in den meisten Fällen die Festzeitsteuerung sein. Diese Stufe ermöglicht den uhrzeitabhängigen automatischen Signalprogrammwechsel für Gruppen- und Einzelkreuzungen mit der Wochenzeitautomatik.

10.2. Verkehrsabhängige Signalprogrammauswahl

Die Festzeitsteuerung ist nur dann ein befriedigendes Verfahren, wenn die Verkehrsmengen immer mit annähernd gleicher Stärke zur gleichen Uhrzeit auftreten.
Zufällige Schwankungen, verursacht durch Unfälle, Bauarbeiten, Schlechtwettereinbrüche, Veranstaltungen und Feiertage, müssen durch adaptive Steuerverfahren gesondert behandelt werden, um einen optimalen Verkehrsfluß im Netz aufrechtzuerhalten.
Der Verkehr wird vom Rechner in Form von Zählwerten über Detektoren erfaßt.

Zufällige Schwankungen werden über ein Ausgleichsverfahren geglättet, so daß festgestellt werden kann, ob die Verkehrsbelastung in auf- oder absteigender Richtung tendiert. Auf Grund der an charakteristischen Punkten im Regelgebiet gewonnenen Meßwerte bekommt der Rechner Aufschluß über die momentane allgemeine Verkehrslage, die durch eine ganz bestimmte Verkehrssituation definiert ist. Vom Rechner wird über ein spezielles Situationsauswahlprogramm für das gesamte Regelungsgebiet festgestellt, ob beispielsweise Morgen-, Normal-, Nachmittag-, Nacht- oder Ausflugsverkehr vorliegt. Da für die einzelnen Regelungsbezirke (Gruppen oder Kreuzungen) diese pauschale Aussage nur bedingt zutrifft, wird für jede verkehrsabhängige Einheit, ausgehend von der Gesamtsituation , noch mal getrennt die eigentliche Signalprogrammauswahl durchgeführt. Diese bestimmt das endgültige Signalprogramm, das der Verkehrssituation auf lokaler Basis am besten gerecht wird.

10.3. Die verkehrsabhängige Signalprogramm-Modifizierung

Das ausgewählte Signalprogramm kann immer nur für eine bestimmte Verkehrssituation , die durch den Belastungsfall charakterisiert ist, zutreffend sein. Kurzfristige Schwankungen, die außerhalb der Grenzen des Belastungsfalles liegen, werden durch Signalprogrammodifizierung ausgeglichen.

Das bei den Verkehrsrechners realisierte Verfahren gliedert sich in drei abgeschlossene Teile bzw. Programme:

1. Meßwerterfassung und Aufbereitung
2. Entscheidungslogik
3. Signalzeitenplanmodifizierung.

Über die Meßwerterfassung wir in jeder Sekunde der anstehende Verkehr erfaßt. Die Meßwertkriterien können vollkommen unterschiedlich sein, wie beispielsweise die Zeitlücke, Verkehrsdichte, Staulänge, der Belegungsgrad u.s.w. Ebenso können die Meßwerte von Knotenpunkt zu Knotenpunkt zu verschiedenen Zeiten in unterschiedlicher Form eingehen und aufbereitet werden. Das pro Sekunde gewonnene Bild stellt den

augenblicklichen Zustand des Verkehrs dar, der im Programm durch eine Zustandsgleichung festgehalten wird. Diese wird an die Entscheidungslogik weitergegeben, die feststellt, welcher Zustand vorliegt und welche Art von Eingriff durchgeführt werden muß.

Dieser kann beispielsweise sein:

1. Verlängern oder Verkürzen von Signalisierungszuständen (Phasen)
2. Verändern von Signalzeiten
3. Einfügen und Ausblenden von Signalen
4. Abwicklung von Anforderungen.

Werden mehrere voneinander abhängige Kreuzungen nach diesem Verfahren gesteuert, so werden Zustands- oder Modifikationsmerkmale untereinander ausgetauscht, so daß eine gegenseitige Beeinflussung, falls notwendig, ebenfalls realisiert werden kann.

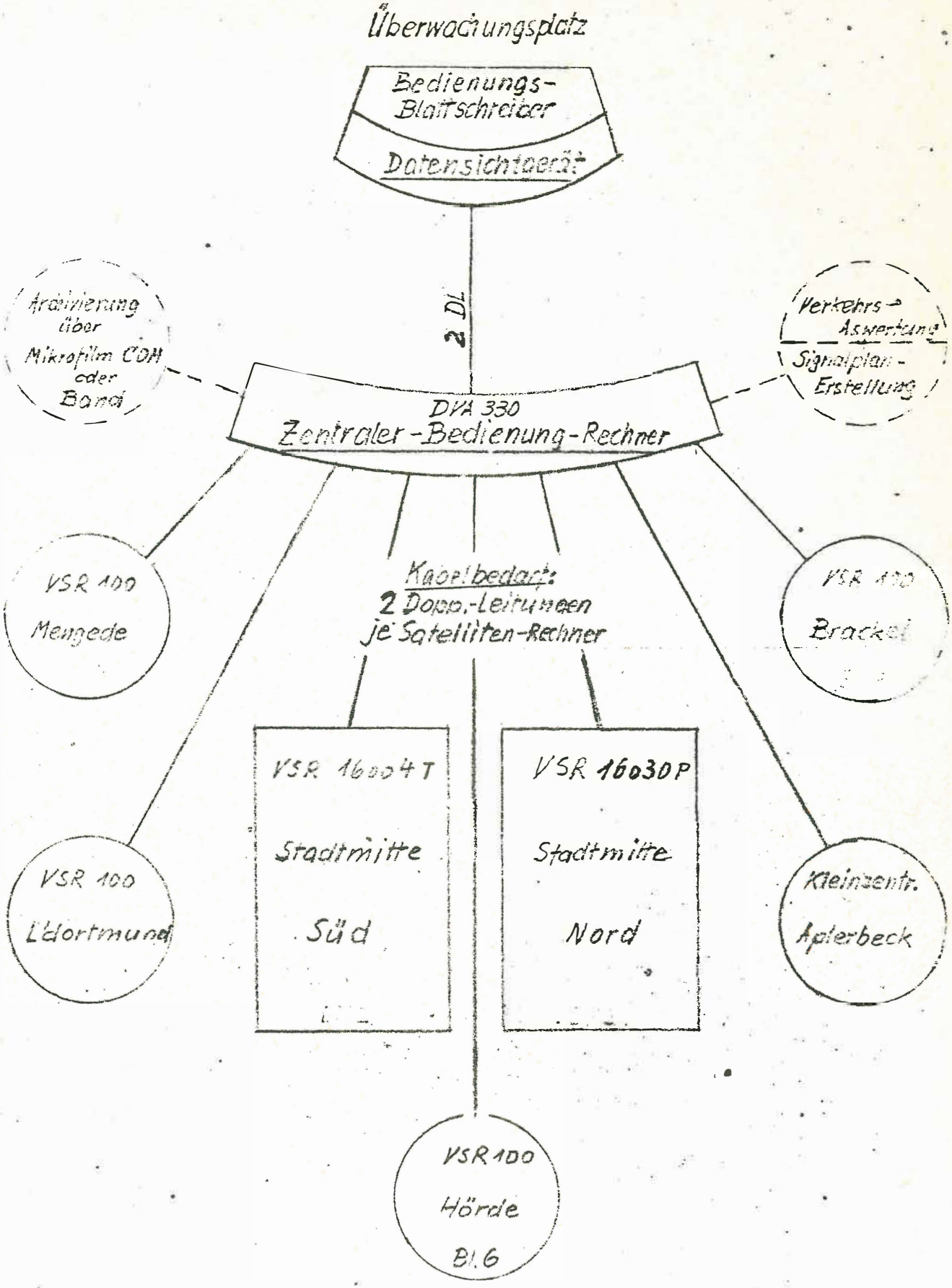
Durch die freie Wahl der Möglichkeiten, die dieses Konzept bietet, kann jede gestellte Aufgabe bzw. jedes beliebige Verfahren im Rahmen einer kurzfristigen Verkehrsabhängigkeit realisiert werden.

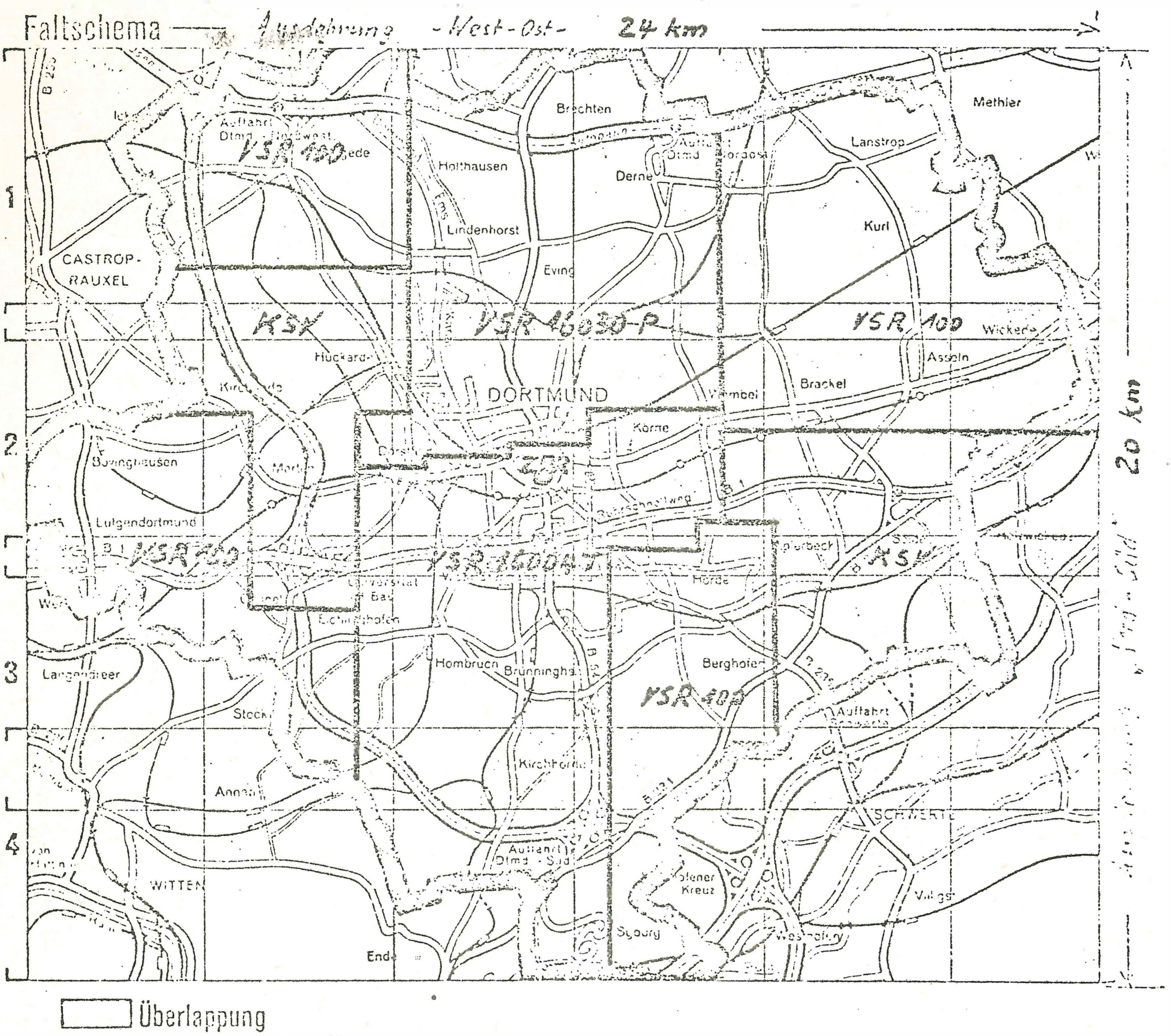
11. Ausgeführte Anlagen

Am Beispiel der Stadt Dortmund möchten wir Ihnen ein ausgeführtes Anlagensystem vorstellen. Im Stadtgebiet Dortmund sind zur Zeit aus der Systemgruppe 304 ein Rechner mit Trommelspeicher und aus der Systemgruppe 330 ein Rechner mit Plattenspeicher sowie ein zentraler Bedienungsrechner eingesetzt. Diese Rechner werden ergänzt in den Vororten durch Kleinrechner mit Mikroprozessoren, die ebenfalls in das zentrale Steuerungssystem einbezogen sind. Die einzelnen Rechner sind über zwei Doppelleitungen mit dem zentralen Bedienungsrechner (ZBR) verbunden. Dem zentralen Bedienungsrechner ist ein Überwachungsplatz zugeordnet, der ebenfalls über zwei Doppelleitungen angeschlossen ist. Der gesamte Datenfluß zwischen den Straßenverkehrssignalanlagen (SVA) und dem Überwachungsplatz wird vom zentralen Bedienungsrechner durchgeführt.

Die Signalisierungszustände der Straßenverkehrssignalanlagen werden mit Hilfe eines Blattschreibers am Überwachungsplatz dokumentiert. Störungsmeldungen werden außerdem über eine Datensichtstation optisch angezeigt. Anstelle des Blattschreibers kann bei Bedarf eine leistungsfähigere Datenausgabeeinrichtung eingesetzt werden, z.B. ein Schnelldrucker.

Der ZBR ermöglicht ein für alle angeschlossenen Rechnersysteme gleichen Bedienungsformalismus am Überwachungsplatz.


Überwachungsplatz
Bedienungs-
Blattschreiber
Datensichtgerät
2 DL
Archivierung
über
Mikrofilm COM
oder
Band
Verkehrs-
Auswertung
Signalplan-
Erstellung
DVA 330
Zentraler-Bedienung-Rechner
Kabelbedarf:
2 Doppel-Leitungen
je Satelliten-Rechner
VSR 100
Mengede
VSR 100
Brackel
VSR 1600 4T
Stadtmitte
Süd
VSR 1600 3DP
Stadtmitte
Nord
VSR 100
Dortmund
Kleinzentr.
Aplerbeck
VSR 100
Hörde
Bl. 6


Faltschema
Ausdehnung -West-Ost- 24 km
20 km
1
2
3
4
VSR 100
KSV
VSR 16030-P
VSR 100
VSR 100
VSR 16004 T
KSV
VSR 400
DORTMUND
CASTROP-RAUXEL
Brechten
Holthausen
Lindenhorst
Eving
Derne
Lanstrop
Methler
Kurl
Wickede
Asseln
Brackel
Körne
Bövinghausen
Lütgendortmund
Langendreer
Hombruch
Brünninghsn
Berghofen
Hörde
Kirchhörde
Annen
WITTEN
SCHWERTE
Syburg
Auffahrt
Überlappung

Literatur:

Das neue Verkehrsrechner-System 16 000 - VSR 16 002, VSR 16 004 -	Bestell-Nr.	D-337/7069
Siemens-Verkehrsrechner-System 16 000 - VSR 16 020, VSR 16 030 -	"	D-337/7749
Grünlicht, Ausgabe 1/75	"	D-337/7107
" , " 2	"	D-337/7113
" , " 5	"	F-337/7116

Bernd REESE, Heiko KRUMM
UNIVERSITÄT KARLSRUHE
Institut für Informatik III
Lehrstuhl für Prozeßinformatik

PRINT - Prozeßinterpreter

Übersicht:

1. Prozeßprogrammiersprachen

PRINT ist eine Prozeßprogrammiersprache, deren Eigenschaften auf heutige Mini- und Mikro-Rechner abgestimmt ist.
Um einen Vergleich mit gängigen Prozeßprogrammiersprachen ermöglichen zu können, sollen zunächst die wichtigsten Eigenschaften der drei Sprachen Realzeit-Basic, Realzeit-Fortran und Pearl aufgeführt werden.

1.1 Realzeit-Basic

Realzeit-Basic ist auf fast allen Mini- und vielen Mikrorechnern realisiert. Dies ist möglich, da geringe Systemgrößen (10-15 Kw) bei der Implementierung auf diesen Rechnern erreicht werden.
Basic ist sehr leicht erlernbar, dies wird weitgehend durch den Dialogbetrieb beim Programmaufbau und durch die einfache Sprachstruktur unterstützt. Die einfache Sprachstruktur erschwert allerdings den Aufbau hierarchisch gegliederter Programme. Insbesondere ist die Sprunganweisung (goto) nicht vermeidbar.
Für den Aufbau von Unterprogrammen existieren primitive Sprachelemente, Parameterübergabe bei Unterprogrammen ist nicht möglich.
In Basic stehen lediglich die beiden Datentypen Real und String zur Verfügung. Andere in der Prozeßdatenverarbeitung häufig gebrauchte Datentypen müssen über diese Datentypen simuliert werden.
Zur Zeit existiert noch kein einheitlicher Standard für Prozeßfunktionen.

1.2 Realzeit-Fortran

Fortran ist eine Programmiersprache, die speziell für technisch wissenschaftliche Anwendungen konzipiert wurde. Das Sprachniveau entspricht nicht mehr den heute üblichen Ansprüchen an eine höhere Programmiersprache.
Durch folgende einfache Zusätze und Erweiterungen wurde Fortran an die Erfordernisse der Prozeßdatenverarbeitung angepaßt:

- Möglichkeit zur Binärdatenverarbeitung über den Typ Integer (AND, OR, XOR sind auf den Datentyp Integer anwendbar, Zugriff auf einzelne Bits einer Integergröße möglich).
- Formaterweiterungen bezüglich Uhrzeit
- Die Prozeßperipherie wird über standardisierte externe Unterprogramme angesprochen
- Aufträge an die Ablaufsteuerung des Betriebssystems, wie z.B. zyklischer Start und Beenden eines Programms, werden ebenfalls durch standardisierte externe Unterprogramme weitergegeben
- Der Verkehr mit der Standardperipherie und der Dateiverwaltung wurde über entsprechende Unterprogramme standardisiert

Für spezielle Betriebssystemaufrufe werden von den Herstellern systemspezifische Unterprogramme zur Verfügung gestellt. Damit ist Fortran, allerdings auf Kosten der Homogenität, eine wirkungsvolle Prozeßprogrammiersprache.

1.3 Pearl

Im Gegensatz zu Realzeit-Basic und Realzeit-Fortran wurde Pearl schon von vornherein als Prozeßprogrammiersprache konzipiert. Durch die Aufteilung der Pearlprogramme in einen Systemteil und einen Problemteil wird eine weitgehende Maschinenunabhängigkeit erreicht. Anwenderprogramme lassen sich modular aufbauen. Die strukturierte Programmierung wird durch entsprechende Sprachkonstruktionen unterstützt. Alle für die Realzeitprogrammierung erforderlichen Datentypen (Binär, Real, Integer, Uhrzeit, Zeitdauer, Verbunde) sind vorgesehen.
Sprachelemente für

- Definition von Tasks (Prozesse)
- Synchronisation
- Kommunikation von Standard- u. Prozeßperipherie
- Interruptbehandlung
- Ablaufsteuerung

stehen zur Verfügung.

Da bei der Konzipierung von Pearl weitgehend keine Rücksicht auf vorhandene Betriebssysteme genommen wurde, ist eine umfangreiche Anpassung existierender Betriebssysteme an die Anforderungen der Sprache erforderlich. Dieser Grund und nicht zuletzt das hohe Sprachniveau lassen eine effektive Realisierung auf Mini- und Mikrorechnern fraglich erscheinen.

2. PRINT-Sprachbeschreibung

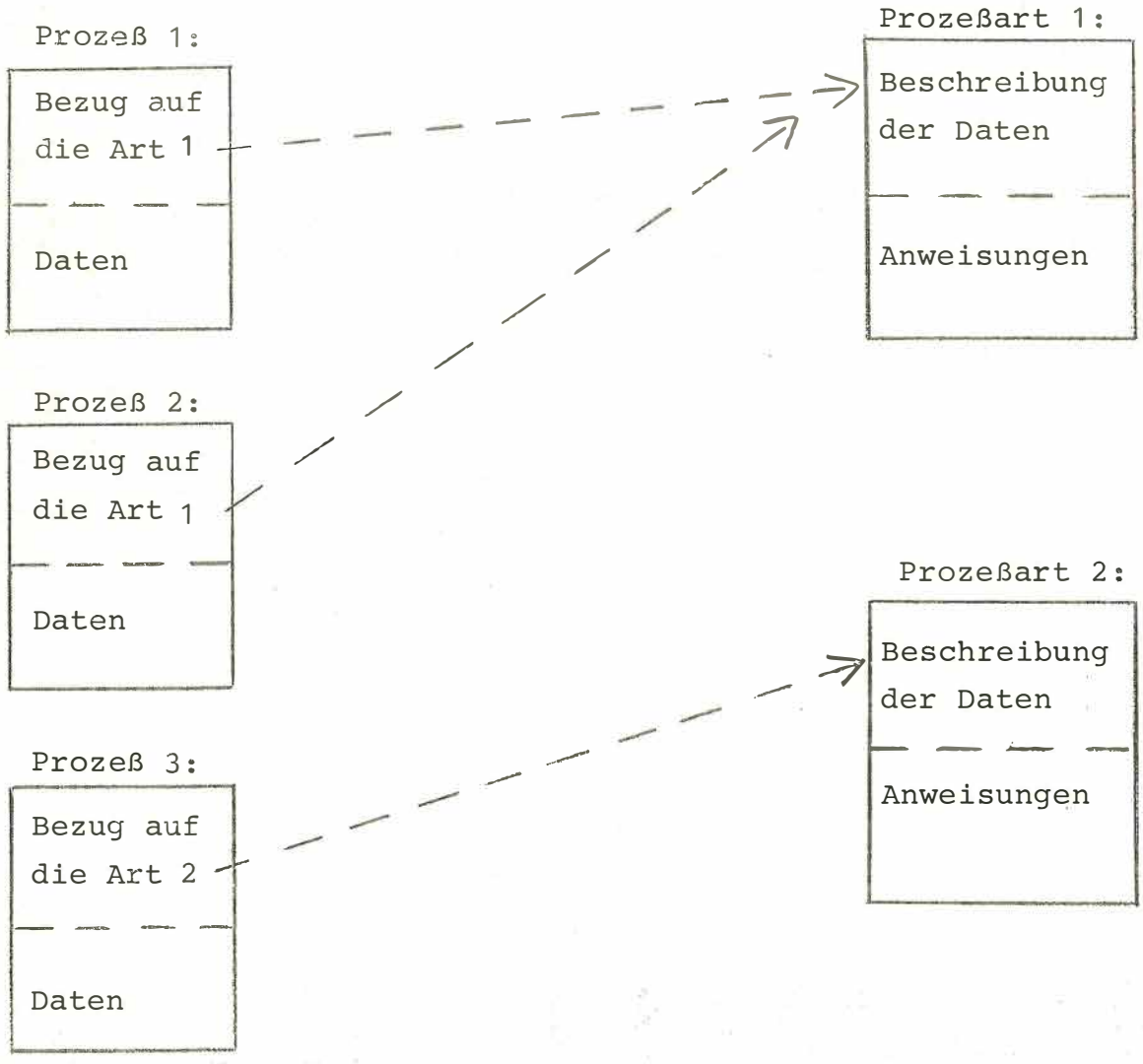
Die Sprache PRINT enthält alle notwendigen Sprachkonstruktionen einer höheren Prozeßprogrammiersprache. Trotzdem läßt die Wahl der Sprachelemente eine kompakte und effektive Realisierung auch auf kleineren Rechnern (Minis, Mikros) zu.
Dies ist insbesondere dadurch erreicht worden, daß parallel zur Sprachentwicklung bereits das dazugehörige Laufzeitsystem mit Realzeitfunktionen konzipiert wurde. Das Laufzeitsystem orientiert sich an der Bytestruktur heutiger Mini- und Mikrorechner. Außerdem wurde die Sprache PRINT so angelegt, daß Anwenderprogramme mit Hilfe eines Programmiersystems ähnlich wie bei BASIC im Dialog erstellt und verändert werden können. Insgesamt erlaubt PRINT problemnahes Programmieren, komfortable Programmerstellung und effektive Programmausführung auch bei Rechnern mit kleiner Systemkonfiguration.
Im folgenden sollen die wesentlichen Sprachelemente der Prozeßprogrammiersprache PRINT aufgeführt und erläutert werden.

2.1 Prozesse

Technische Prozesse werden in der Regel durch ihre Zustandsgrössen und deren dynamisches Verhalten beschrieben. Komplexere technische Prozesse werden durch unabhängige Teilprozesse und deren Zusammenwirken dargestellt.
In ähnlicher Weise sollten sich in einer höheren Prozeßprogrammiersprache Prozesse softwaremäßig realisieren lassen.
Zur Definition von Prozessen steht in PRINT folgende Sprachkonstruktion zur Verfügung:

```
PROCESS    <name>
  [Deklarationsteil
  [Anweisungsteil
END
```

Im Deklarationsteil werden die Zustandsgrößen des Softwareprozesses beschrieben; der Anweisungsteil beschreibt den funktionellen Ablauf des Prozesses.
Eine derartige Prozeßdefinition dient als Muster zur Erzeugung von einem oder mehreren Prozessen dieser Art. Jeder erzeugte Prozeß besteht aus einem Datenbereich und einem Bezug auf die definierte Art, in der die Daten beschrieben werden, und die zugehörige Anweisungsfolge enthalten ist.
Im nachstehenden Bild existieren zwei Prozesse der Prozeßart 1 und ein Prozeß der Prozeßart 2:



2.2 Zusammengesetzte Prozeßarten:

Da es häufig vorkommt, daß Prozeßarten gemeinsame Anweisungsfolgen und/oder gemeinsame Datendeklarationen besitzen, ist es möglich, gemeinsame Teile verschiedener Prozeßarten zu einer gemeinsamen Oberprozeßart zusammenzufassen und damit nur einmal zu programmieren.
Im folgenden Beispiel werden drei Prozeßarten U1, U2 und O beschrieben. O ist Oberprozeßart zu den Arten U1 und U2:

```
PROCESS O
[
[
END
PROCESS U1 OF O
[
[
END
PROCESS U2 OF O
[
[
END
```

2.3 Startprozeß

Jedes aus mehreren Prozeßarten bestehende Prozeßsystem muß eine Prozeßart mit der Bezeichnung MAIN enthalten. Die Bearbeitung eines Prozeßsystems beginnt mit der automatischen Erzeugung und dem Start eines Prozesses dieser Art.
Hiermit kann die Prozeßart MAIN die Rolle eines Hauptprogramms übernehmen. Alle Daten und Prozeduren des Startprozesses stehen den übrigen Prozessen zur Verfügung.

```
PROCESS MAIN
[
[
END
:
weitere Prozeßarten
:
```

2.4 Datentypen

Folgende in der Prozeßdatenverarbeitung notwendige Datentypen stehen zur Verfügung:

Integer	(Wortbreite: 16 Bit)
Real	(8 Stellen Genauigkeit durch Tetradendarstellung)
Byte	(8 Bit, für Zeichen, Binärgrößen und Zeichenketten (Strings))
Uhrzeit	(Stunde, Minute, Sekunde, Millisekunden)
Zeitdauer	(Stunden, Minuten, Sekunden, Millisekunden)
Prozeßmenge	(erlaubt den Bezug auf einen oder mehrere Prozesse)

Beispiele für Deklaration von Variablen
(Deklaration von Variablen ist obligatorisch):

INT I,J,K	Die Integervariablen I,J,K werden vereinbart
REAL(1ØØ,2ØØ)WERT	Vereinbarung eines zweidimensionalen Realfeldes (Grenzen 1ØØ und 2ØØ) mit dem Bezeichner WERT
BYTE(L) T1, T2	Dynamische Feldvereinbarung zweier eindimensionaler Bytefelder T1 und T2
CLOCK CL	Uhrzeitvariable CL
DUR D1, D2	Zeitdauervariablen D1, D2
SET REGLER	Deklaration einer Prozeßmengenvariablen mit dem Bezeichner REGLER

Beispiele für Konstantendarstellung:

Integer:	3, 1ØØ, 32767	natürliche Darstellung
	B'1ØØ1Ø1'	Binärkonstante
	O'177'	Oktalkonstante
	H'AEØØ'	Hexadezimalkonstante
Byte:	'A'	Zeichen
	'(255)','(Ø)','(1)'	Dezimaläquivalent
	'EIN TEXT'	Text
	'+-(39,1,2)*/'	Text mit Dezimaläquivalent
Real:	3.14	
	1.5 E-12	
Uhrzeit:	C'12:Ø:Ø'	12 Uhr
	TIME	Systemkonstante, aktuelle Zeit
Zeitdauer:	D'Ø:Ø:1:3Ø'	1 Sekunde 3ØØ Millisekunden
Prozeßmengen (Systemkonstanten):		
	THIS	Bezug auf aktuellen Prozeß
	NONE	Leermenge

2.5 Zuweisungen, Ausdrücke, Operatoren:

Die Schreibweise von Zuweisungen und Ausdrücken entspricht im wesentlichen der allgemein üblichen:

```
R:= X*(A+X*(B+X*C))+D
RFELD(I,J):= INTR(I+J)/10.0
```

Zusätzlich besteht die Möglichkeit, mehrere Elemente eines Feldes in einer Anweisung gleichzeitig anzusprechen.

```
TEXT(3:*):= 'XYZ', T1(1:10)
```

Text (3:*) identifiziert einen linearen Bereich des Feldes TEXT ab Element 3 mit variabler Länge.
In diesen Bereich werden die Werte der Ausdrucksliste der rechten Seite sequentiell übertragen.
Hier am Beispiel die Zeichen X, Y und Z und die ersten zehn Bytes des Feldes T1.

Monadische Operationen:

ABS SIGN ENTIER - NOT	
INTR	Explizite Artanpassung von Integer nach Real
RINT	" " " Real nach Integer
INTB	" " " Integer nach Byte
BINT	" " " Byte nach Integer
CARD	Anzahl der Elemente einer Prozeßmenge
NEW	erzeugt einen neuen Prozeß

Dyadische Operatoren:

+ - * / MOD * *	Arithmetische Operatoren
OR AND XOR	Logische Operatoren
= <> < > <= >=	Vergleichsoperatoren
CSHIFT LSHIFT	Zyklisches und logisches Schieben
LBIT RBIT	Bitleseoperatoren
CON DIS DIF MEMB	Prozeßmengenoperatoren

Prozeßmengenoperatoren:

Beispiel für das Erzeugen eines Prozesses der Art REGLER mit den Parametern P,I,D und der Priorität 5:

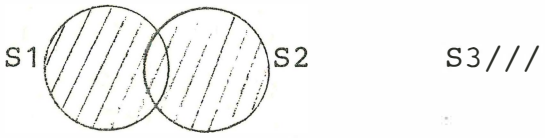
```
R1:= NEW REGLER(P,I,D) PRIO 5
```

Die Prozeßmengenvariable R1 enthält als Wert den Bezug auf den soeben erzeugten Prozeß.

Die Prozeßmengenoperatoren CON, DIS, DIF und MEMB sollen anhand von Bildern erläutert werden:

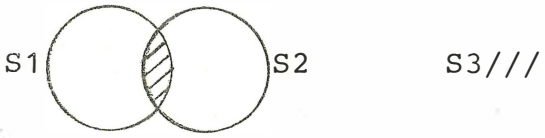
CON: Vereinigung zweier Mengen

S3:=S1 CON S2



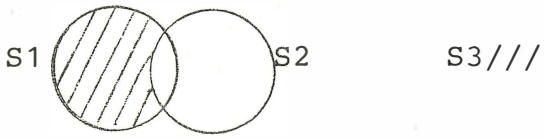
DIS: Schnittmenge

S3:=S1 DIS S2



DIF: Differenzmenge

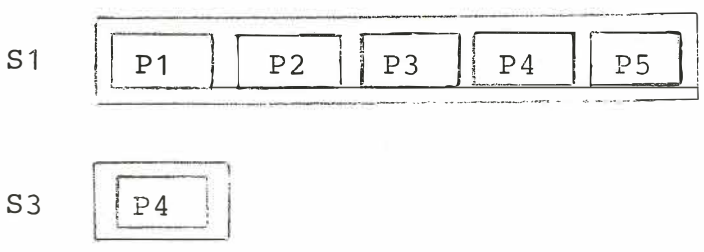
S3:=S1 DIF S2



MEMB: Indizierungsoperator

Eine Prozeßmenge wird intern als lineare Liste ihrer Elemente dargestellt, mit dem MEMB Operator kann auf einzelne Elemente dieser Liste Bezug genommen werden.

S3:=S1 MEMB 4



2.6 Kontrollanweisungen

```
IF A=B  AND  C>1ØØ THEN          IF-Anweisung
  [
ELSE
  [
FI
```

```
I:=1
WHILE I<=1ØØ DO                  While-Anweisung
  FELD(I):=I**2                  (Test am Schleifenanfang)
  I:=I+1
DONE
```

```
DO                               Until-Anweisung
[                                (Test am Schleifenende)
UNTIL I=1ØØ
```

```
DO                               Endlosschleife
[
DONE
```

```
CASE I                           Verteiler
[
NEXT
[
NEXT
 :
OUT
[
ESAC
```

2.7 Unterprogramme

Deklaration und Anlauf von Unterprogrammen soll am folgenden Beispiel zur Fakultätsberechnung erläutert werden:

```
20    PROCESS MAIN
40     INT I
60     SUB FACULTAET(ARG,ERG)
80      INT ARG,ERG
100     NAME ERG
120     IF ARG<=1 THEN
140      ERG:=1
160      ELSE
180      CALL FACULTAET(ARG-1,ERG)
200      ERG:=ERG*ARG
220      FI
240     RETURN

260    COM -------- AUFRUF DES UNTERPROGRAMMS ----------
280    CALL FACULTAET(10,I)
300    END
```

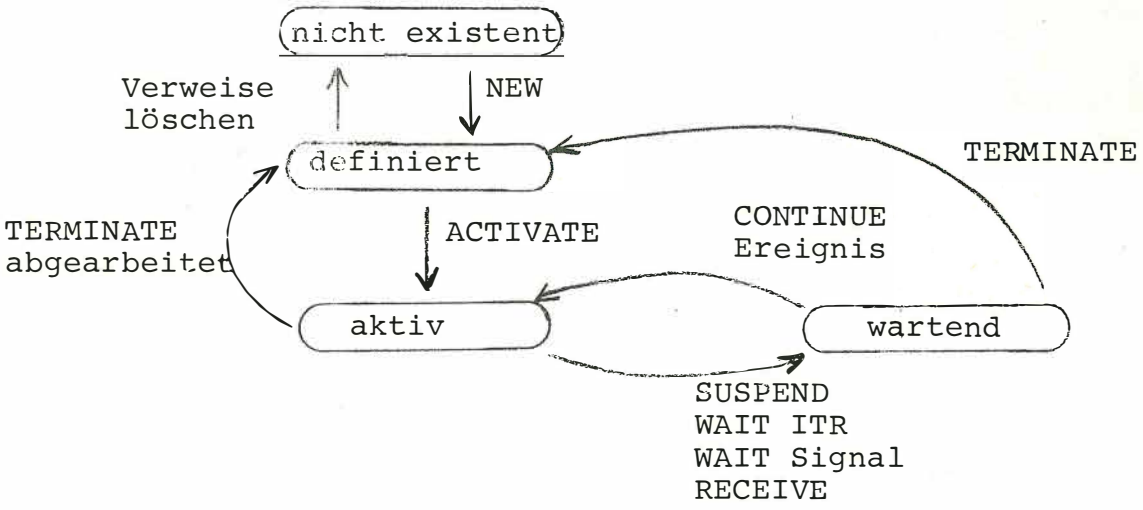
Das Unterprogramm wird durch die Anweisungen SUB und RETURN eingeschlossen. Parameter können per value oder per name übergeben werden.
Rekursiver Aufruf von Unterprogrammen ist möglich.
Unterprogramme werden mit Hilfe der CALL Anweisung aufgerufen und mit den aktuellen Parametern versehen.

Übergabe von Feldern an Unterprogramme:
Beispiel:

```
INT(1ØØ,2ØØ)FELD
:
SUB UP(F)
 INT(,)F
 :
RETURN
:
CALL UP(FELD)
```

2.8 Prozeßzustände und Übergänge

Übergangsdiagramm



Prozesse können die Zustände 'nicht existent', 'definiert', 'aktiv' und 'wartend' einnehmen.

Durch Erzeugen eines Prozesses mit dem NEW Operator gelangt er vom Zustand 'nicht existent' in den Zustand 'definiert'.

```
S1:= NEW REGLER(P,I,D) PRIO 5
```

Das Aktivieren eines definierten Prozesses wird durch die ACTIVATE Anweisung bewirkt.

```
ACTIVATE S1
```

Die SUSPEND Anweisung blockiert einen aktiven Prozeß, bis er durch eine CONTINUE Anweisung deblockiert wird.

```
SUSPEND  S1
CONTINUE S1
```

Durchläuft ein Prozeß eine WAIT ITR (Warten auf Interrupt), eine WAIT Signal (Warten auf Signal) oder eine RECEIVE (Warten auf Botschaft und Botschaft übernehmen) Anweisung, blockiert er sich selbst, bis das entsprechende Ereignis (Interrupt, Signal, Botschaft) eintrifft und ihn deblockiert.

WAIT ITR 3	Warten auf Interrupt 3
WAIT S3	Warten auf ein Signal von einem der Prozesse der Menge S3.
SIGNAL S2	Senden eines Signals an alle Prozesse der Menge S2.
RECEIVE SENDER S: A,B,C	Warten auf eine Botschaft, Übernahme der Absenderkennung in der Prozeßmengenvariablen S und der Botschaft in den Variablen A,B,C.
RECEIVE FROM S3: A,B,C	Warten auf eine Botschaft von einem der Prozesse der Menge S3, Übernahme der Botschaft in A,B,C.
SEND A,B,C TO S1	Erzeugen einer Botschaft mit den Werten von A,B,C und absenden an alle Prozesse der Menge S1.

Ist ein Prozeß abgearbeitet, d.h. er hat die letzte END Anweisung erreicht oder wird ein Prozeß durch eine TERMINATE Anweisung explizit terminiert, so gelangt er wieder in den Zustand 'definiert' und kann gegebenenfalls neu aktiviert werden.

```
TERMINATE S1
```

Wenn auf einen definierten Prozeß keine Bezugnahme mehr möglich ist, weil er in keiner Prozeßmenge des Systems mehr Mitglied ist, so wird er gelöscht und die von ihm belegten Speicherbereiche werden freigegeben.

Einplanungen

Die Anweisungen ACTIVATE, TERMINATE, SUSPEND, CONTINUE, PREVENT, SIGNAL können wie in den vorigen Beispielen sofort ausgeführt werden oder erst zu einem angegebenen Zeitpunkt nach einer angegebenen Zeitdauer oder bei Eintreten eines Interrupts:

```
ACTIVIATE S1 AT C'12:Ø:Ø'
TERMINATE S1 AFTER DAUER
CONTINUE  S1 ONITR 3
```

Durch die DELAY Anweisung können Prozesse bis zu einem Zeitpunkt oder über eine Zeitdauer blockiert werden.

```
DELAY   THIS  UNTIL C'17:Ø:Ø'
DELAY   S1    DURING D'Ø:1Ø:Ø'
```

Einplanungen können durch die PREVENT-Anweisung zurückgenommen werden.

Interruptanweisungen:
Interrupts können maskiert, demaskiert und per Anweisung simuliert werden:

```
DISABLE  5
ENABLE   I+1
TRIGGER  4
```

2.9 Externaufrufe

Über Externaufrufe können Assemblerprogrammstücke (Externroutinnen) ausgeführt werden.
Aufruf: EXTERN < Nr. der Routine >
Zum Beispiel kann die Digitalausgabe oder die Analogeingabe über Externroutinen angesprochen werden.

Beispiel für Digitalausgabe:

```
SUB  DIGAUS(WERT)
  INT WERT
  EXTERN 1
 RETURN
  :
CALL  DIGAUS(H'Ø5A1')
  :
```

Beispiel für Analogeingabe:

```
SUB  ANEIN(WERT)
 INT  WERT
 NAME WERT
 EXTERN 2
 WAIT ITR 4
 EXTERN 3
RETURN
:
CALL  ANEIN(J)
:
```

Für die Standard- und Prozeßperipherie werden dem Benutzer genormte PRINT-Treiberprozesse und Subroutinen zur Verfügung gestellt, er kann sie aber nach Bedarf erweitern und verändern.

2.10 Beispiel für ein einfaches Prozeßsystem

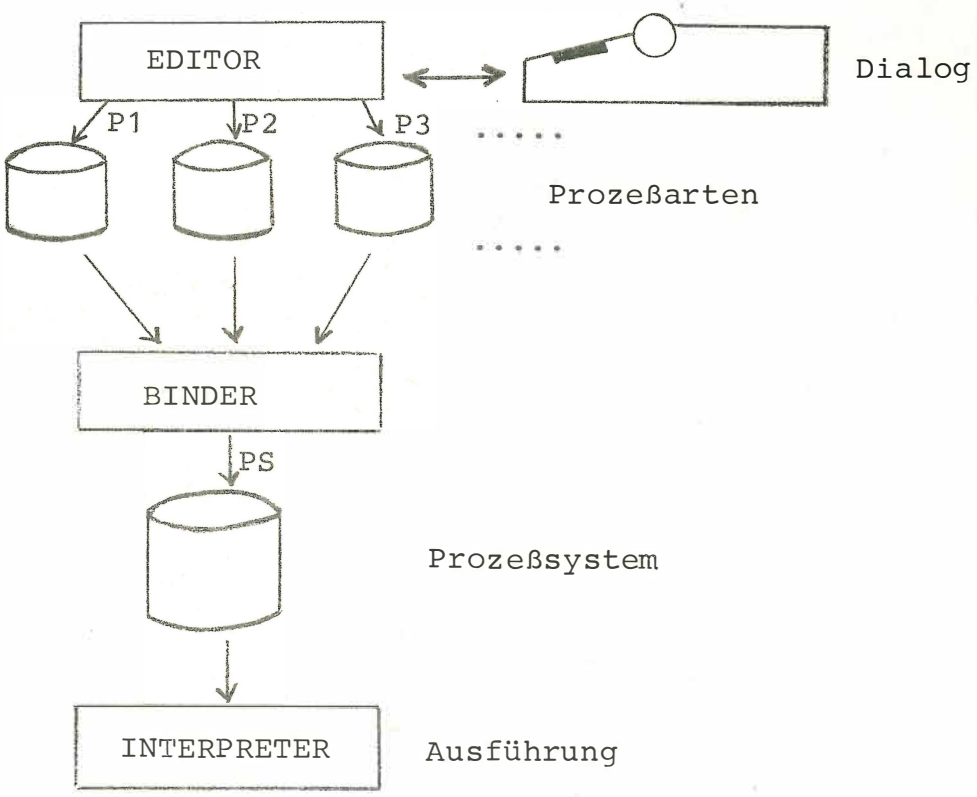
```
20    PROCESS MAIN
40      SET R
60      R:=NEW REGLER(D'0:0:1:0',6,7)
80      R:=R CON NEW REGLER(D'0:0:0:30',4,5)
100     ACTIVATE R AT C'15:0:0:0'
120     TERMINATE R AT C'17:0:0:0'
140     END


20    PROCESS REGLER(ZYKL,AS1,AS2)
40      DUR ZYKL
60      INT AS1,AS2
80      DO
100
120       COM <--- REGELALGORITHMUS --->
140
160       DELAY THIS DURING ZYKL
180       DONE
200     END
```

Der Prozeß REGLER führt einen Regelalgorithmus zyklisch mit der Zykluszeit ZYKL aus. Die Parameter AS1 und AS2 sind Schnittstellenparameter. Der Regelalgorithmus selbst ist nicht weiter ausgeführt.
Im Prozeß MAIN werden zwei Prozesse der Art REGLER mit unterschiedlichen Zykluszeiten und Schnittstellenparametern erzeugt und zu einer Prozeßmenge vereinigt. Die Prozesse dieser Menge, d.h. die beiden Regelprozesse, werden um 15.00 Uhr gestartet und um 17.00 Uhr beendet.

3. Implementierung des PRINT-Systems

3.1 Programme des Systems



3.1.1 EDITOR

Die Aufgabe des EDITOR-Programmes ist die Prozeßarterstellung im Dialog. Die Prozeßart wird zeilenweise mit Angabe der Zeilennummer aufgebaut. Jede eingegebene Zeile wird sofort auf lokale syntaktische Richtigkeit überprüft und in einen 'Analysecode' übersetzt. Es ist möglich, Zeilen gemäß Zeilennummer einzuordnen, zu überschreiben (verändern) und zu löschen. Einzelne Zeilen, Bereiche oder das ganze Programm können jederzeit am Bediengerät oder Schnelldrucker aufgelistet werden. Die Anweisungen können über das RENUM-Kommando neu nummeriert werden. Mit dem SAVE-Kommando werden Prozeßarten auf Datei abgespeichert. Hierbei wird die Struktur der Prozeßart überprüft. Auf Datei befindliche Prozeßarten können jederzeit über das OLD-Kommando wieder eingelesen und dann erneut bearbeitet werden.

3.1.2 BINDER

Die Aufgabe des Binders ist das Zusammenfügen von einzelnen Prozeßarten zu einem zusammenhängenden Prozeßsystem.
Der Binder erzeugt interpretierbaren Code.
Hierbei werden letzte semantische Überprüfungen, die nur im Zusammenhang möglich sind, durchgeführt.
Beispiel für eine Bedienung:

```
LINK  P1, P2, P3 : PS
```

3.1.3 INTERPRETER

Die Aufgabe des Interpreters ist die Interpretation des vom Binder erzeugten Codes.
Im groben läßt sich der Interpreter in folgende Funktionen aufteilen:

- Anweisungsinterpretation
- Ausdrucksinterpretation
- Organisationsteil mit
  - Hauptspeicherverwaltung
  - Interruptbearbeitung
  - Auftragsverwaltung und Bearbeitung
  - Synchronisation und Kommunikation

Da der Interpreter einen eigenen Organisationsteil besitzt, entfällt die komplizierte Anpassung an vorhandene Betriebssysteme, bzw. kann er ohne großen Aufwand als Stand-alone-System betrieben werden.

3.2 Implementierungsverfahren

Das PRINT-System wurde zunächst auf dem Großrechner UNIVAC 1108 in der höheren Programmiersprache SIMULA 67 implementiert.
Mit Hilfe dieses Systems (Editor, Binder, Interpreter in Simula) werden Editor und Binder in der Sprache Print geschrieben. Damit stehen Editor und Binder in maschinenunabhängiger Form zur Verfügung (Portabilität).
Danach wurde der Interpreter mit Organisationsteil in der Assemblersprache 300 auf dem Prozeßrechner S330 der Firma Siemens implementiert. Da Editor und Binder bereits in maschinenunabhängiger Form vorlagen, waren sie sofort auf diesem Interpreter ablauffähig.
Zur Zeit werden Interpreter für folgende Maschinen realisiert:
S330, S310, PDP11, Texas 960.