KFA

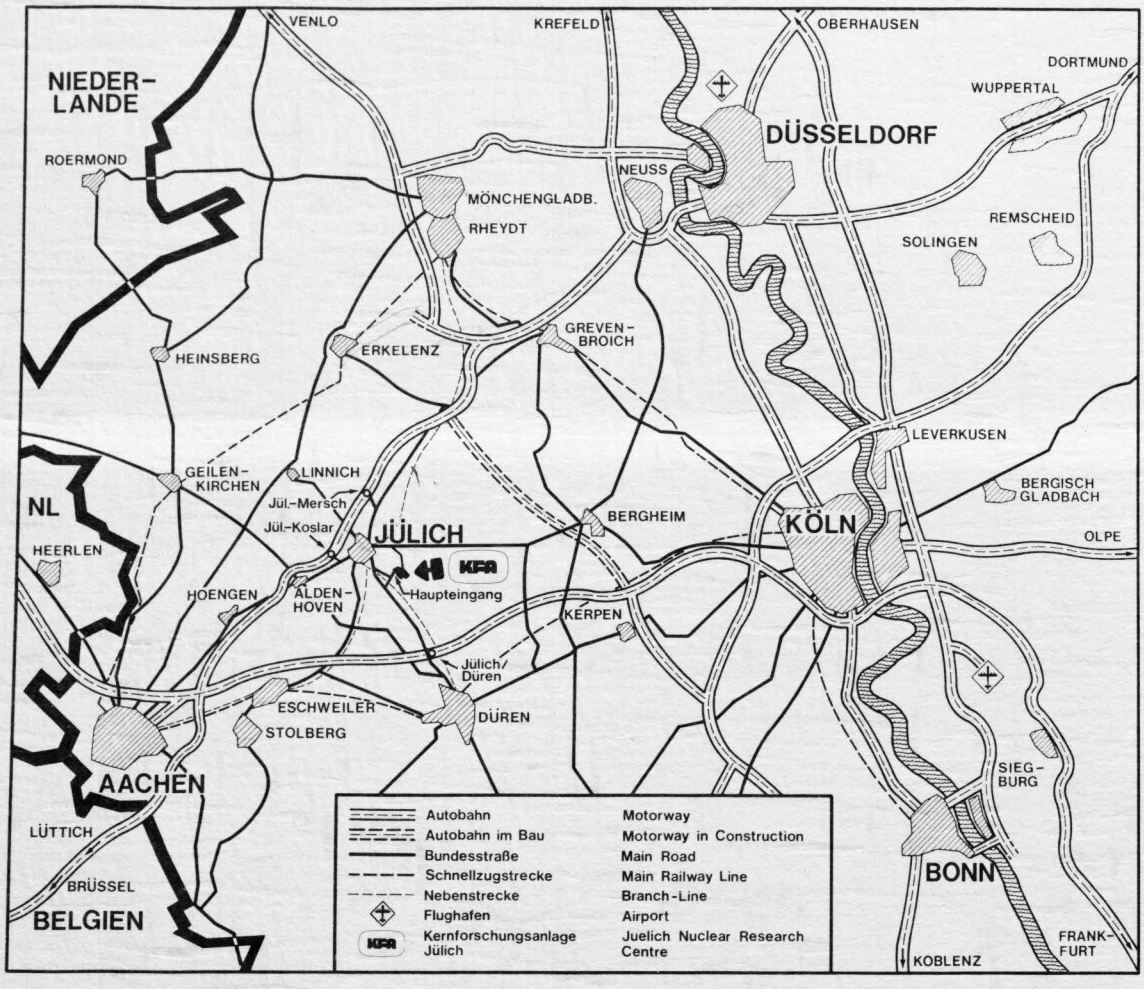
KERNFORSCHUNGSANLAGE JÜLICH GmbH

Tagungsbericht der 11. Jahrestagung des Siemens Prozeßrechner Anwenderkreises I vom 12. bis 14. Mai 1980 in der Kernforschungsanlage Jülich

Herausgeber:

Erhard Bachner

Jül - Conf - 39
September 1980
ISSN 0344-5798(Jül-Conf-Bericht)
ISSN 0171-2950
(Tagungsbericht SAK)



Als Manuskript gedruckt

Berichte der Kernforschungsanlage Jülich - Jül - Conf - 39

Zu beziehen durch: ZENTRALBIBLIOTHEK der Kernforschungsanlage Jülich GmbH
Postfach 1913 · D-5170 Jülich 1 (Bundesrepublik Deutschland)
Telefon: (02461) 61-0 · Telex: 833556 kfa d

Tagungsbericht der 11. Jahrestagung des Siemens Prozeßrechner Anwenderkreises I vom 12. bis 14. Mai 1980 in der Kernforschungsanlage Jülich

Herausgeber:

Erhard Bachner

Inhaltsverzeichnis

Vorwort

Vom 12. bis 14. Mai 1980 fand in der Kernforschungsanlage Jülich die 11. Jahrestagung des Siemens-Prozeßrechner-Anwenderkreises I (SAK I) statt. Es nahmen 129 Anwender und Mitarbeiter von Ausbildungs- und Forschungsstätten sowie von der Industrie, hier insbesondere von Siemens, teil.
In 19 Vorträgen wurde über neue Arbeiten und Erfahrungen auf dem Software-Sektor, aber auch über Hardware-Entwicklungen berichtet. Von der Firma Siemens wurde über neue und geplante Entwicklungen informiert. Im übrigen konnte der fruchtbare Erfahrungs- und Informationsaustausch unter den Anwendern fortgesetzt werden.

Der vorliegende Bericht enthält alle auf der Tagung gehaltenen Vorträge, soweit sie von den Referenten zur Verfügung gestellt wurden. Allen Autoren sei an dieser Stelle herzlich gedankt.

Der Kernforschungsanlage Jülich und dem Hause Siemens sei im Namen des Anwenderkreises für die großzügige Unterstützung herzlich gedankt, die die Durchführung der Tagung und die Herausgabe des Tagungsberichts ermöglichte.

Die 12. Jahrestagung wird 1981 an der Universität Karlsruhe stattfinden.

Erhard Bachner

Tagungsbeauftragter und Herausgeber

Programm

Montag, den 12. Mai 1980

14.00 Uhr	*Eröffnung* K. Heim, SAK-Vorstand
	Begrüßung durch den Vorstand der KFA
	E. Bachner, SAK-Tagungsbeauftragter *Tagungsinformation*
14.30 Uhr	H. Inhoven, KFA Jülich/ZFR Schnelle Datenerfassung aus dem Sicherheitssystem des Reaktors FRJ 2.
15.05 Uhr	E. Bachner, KFA Jülich/ZAT Belegen und Freigeben für ORG K.
15.30 Uhr	Dr. W. Tenten, KFA Jülich/ZEL Experimentorientiertes Rechenzentrum in der KFA
15.45 Uhr	*Kaffeepause*
16.15 Uhr	J. Weiss, H. Widner, TU Wien Standardisierte Anschaltung für graphische Geräte an der Serie 300/16
16.50 Uhr	F.J. Polster, KfK Karlsruhe, IDT FAQUEL: Ein interaktives Datenbank-Anfragesystem
17.30 Uhr	J. Thornton, W. Schmid, E. Pohl, TU Braunschweig Rechnerkopplung zwischen SIEMENS PR 330 und ICL 1906 S
17.50 Uhr	*Schluß*

Dienstag, den 13. Mai 1980

9.00 Uhr	Hochmuth, Dornier System GmbH Erläuterung und Demonstration eines autom. Fahrplan-auskunftsystems mit Ausgabe synthetischer Sprache und Bildschirmtext.
9.30 Uhr	N. Portner, Siemens Erlangen Datenverwaltungssystem DVS 300
10.00 Uhr	P. Fischer, Siemens Karlsruhe SPOOL 300
10.30 Uhr	H. Kinnen, Siemens Erlangen SINEC300 - Rechnerverbund mit den Siemens Systemen 300
11.00 Uhr	*Kaffeepause*
11.30 Uhr	G. Just, Siemens Karlsruhe SIGRIS – System für graphische Datenverarbeitung mit dem Prozeßrechnersystem 300-16
12.00 Uhr	Dr. Vetter, Siemens Erlangen PASCAL 300 – Eine neue Programmiersprache für die Siemens Systeme 300
12.30 Uhr	Dr. Vetter, Siemens Erlangen Leistungsumfang und Einsatzgebiet der zwei FORTRAN-Compiler der Siemens Systeme 300
13.00 Uhr	*Mittagspause*
14.00 Uhr	Besichtigungen
15.30 Uhr	*Kaffeepause*
16.00 Uhr	Diskussion mit der Firma Siemens
17.00 Uhr	SAK-Vollversammlung
18.00 Uhr	*Schluß*
19.00 Uhr	Empfang der Firma Siemens im Hotel „Zum alten Forsthaus", Vossenack

Mittwoch, den 14. Mai 1980

9.00 Uhr	Prof. Dr. K. Weise, PTB Braunschweig Allgemeine Struktur von Programmsystemen für Meßaufgaben; Programmbausteine und Kommando-Interpreter
9.25 Uhr	H.-J. Schuster, K. Weise, PTB Braunschweig Universelle Routinen für Siemens-Prozeßrechner 300 zur Alarmbehandlung in peripherspeicherresidenten Programmen
9.50 Uhr	A. Plewnia, H. Lindemann, PTB Braunschweig Dateitransfer im Dialogmodus
10.15 Uhr	*Kaffeepause*
10.45 Uhr	P. Heine, Fraunhofer-Institut Karlsruhe PEARL-Programmierung auf der PR 310: Betriebssystem
11.05 Uhr	L. Lorenz, Fraunhofer-Institut Karlsruhe PEARL-Programmierung auf der 310: Übersetzungssystem
11.35 Uhr	J. Kippe, Fraunhofer-Institut Karlsruhe PEARL-Programmsysteme zur Lösung umfangreicher Automatisierungsaufgaben
12.00 Uhr	M. Trautner Ein PEARL-Cross-Compiliersystem für eine S 310 und Anwendungen in der Labor-Meßtechnik
12.30 Uhr	T. Neidmann, Daimler-Benz AG MOBEST – Ein Remote-Job-Monitor Bedien- und Steuer-programm im Entwicklungszentrum
13.00 Uhr	*Schluß*

Schnelle Datenerfassung aus dem Sicherheits-System des Reaktors FRJ - 2 (DIDO)

Heinz Inhoven
Zentralabteilung Forschungsreaktoren
Kernforschungsanlage Jülich G.m.b.H.
5170 J Ü L I C H

April 1980

Zusammenfassung

Die Zentralabteilung Forschungsreaktoren der Kernforschungsanlage Jülich betreibt u.a. seit 1962 die Reaktoren FRJ-1 (MERLIN) und FRJ-2 (DIDO). Speziell für die Bedürfnisse des DIDO wurde 1976 ein Rechner SIEMENS 330 in Betrieb genommen, 1978 ein weiterer Rechner gleichen Typs für allgemeine Aufgaben der gesamten ZFR. Aus dem Anwendungsbereich "Datenerfassung und -verarbeitung am FRJ-2" wird im vorliegenden Bericht vor allem der Komplex "Schnelle analoge und binäre Signale" vorgestellt. Schwerpunkte dieses Anwendungsbereiches waren bzw. sind noch: allgemeine Probleme der Adaptierung einer derzeit 14 Jahre alten Reaktoranlage an einen Digitalrechner; i.E.: Daten-Entkopplung im Sicherheits-System des Reaktors, -Erfassung mittels CAMAC, -Transfer über einen "extended branch", Erfassungs-Software als HRP, Zwischen-Speicherung als CD, Interpretations-Software als PRP.

Anmerkung

Die folgende Darstellung ist ein Auszug aus einer gleichnamigen, ausführlichen Veröffentlichung. Auf die dort enthaltenen Details mit zahlreichen Bildern, Diagrammen und Beispielen muß in diesem Rahmen verzichtet werden. Interessenten können die Veröffentlichung unter Nr.1658 bei der Zentralbibliothek der KFA beziehen.

1. Reaktor

Der Reaktor FRJ-2 (DIDO) ist ein mit ca. 11 t Schwerwasser moderierter und gekühlter Forschungsreaktor mit einer thermischen Leistung von 25 MW und einem thermischen Neutronenfluß von max. $2{,}2 \cdot 10^{14}$ $(cm^2 sec)^{-1}$. Sein Kern besteht aus 25 Ringspalt-Brennelementen mit insgesamt ca. 3,5 kg hochangereichertem U-235. Für die experimentelle Nutzung stehen zahlreiche vertikale und horizontale Kanäle mit unterschiedlichen Abmessungen, Neutronenflüssen und -spektren, sowie zylindrische zentrale Bestrahlungseinrichtungen in den Brennelementen zur Verfügung. Die mit Hilfe des FRJ-2 unterstützten Forschungsvorhaben betreffen im wesentlichen Festkörperphysik mit Neutronen, Materialbestrahlungen und Loop-Versuche.

2. Sicherheits-System

Unter Sicherheits-System [nach neuerem Sprachgebrauch: Reaktorschutz-System] sind alle meß- und schaltungstechnischen Komponenten der Reaktoranlage zu verstehen, die der nuklearen und reaktortypischen, konventionellen Sicherheit dienen. Das Sicherheits-System steht zu diesem Zweck mit dem Reaktor und seinen Neben- und Hilfseinrichtungen in Wechselwirkung. In der gesamten Reaktoranlage werden etwa 550 Meßwerte und Zustände größtenteils mittels Kontrollgang, Ablesung und manueller Dokumentation durch das Personal überwacht. Ca. 150 Meßwert- und Zustandsüberwachungen jedoch wirken analog oder binär auf das Sicherheits-System unmittelbar ein. Die logische Verknüpfung der daraus gewonnenen Informationen geschieht in einer Relais-Schaltung konventioneller Bauart; sie verarbeitet auch alle mittelbaren Bedienungs-Eingriffe. Die binären Informationen aus Zustands-, Grenzwert- und Experimente-Überwachung, sowie manuelle Signale münden über einen umfangreichen Rangier-Verteiler in die aus ca. 900 Relais bestehende Auswertungs-Schaltung, beide in einem Raum unterhalb der Reaktorwarte.

Im folgenden Abschnitt wird im Hinblick auf die vorzustellende Datenverarbeitung beispielhaft auf einige Schaltungsdetails des Sicherheits-Systems eingegangen:

2.1 Relais-Logikschaltung

Alle Informationen aus den Zustands- und Grenzwertüberwachungen werden in der Relais-Logikschaltung je nach sicherheitstechnischem Gewicht in verschiedenen, nach ihren Folgefunktionen benannten Kreisen ausgewertet. In der Reihenfolge der Wichtigkeit unterscheidet man Überwachungs-Kriterien für: Schnellabschaltung, Abschaltung mit Hallenschluß, Abschaltung, Steuerblockierung, Regelung, Steuerung, Warnung. Unter Schnellabschaltung ist die schnellstmögliche Leistungsreduzierung mit max. Subkritikalität des Reaktors unter Beteiligung aller verfügbaren Absorber zu verstehen. Jedes Überwachungs-Kriterium ist zweifach in der Logikschaltung vertreten, nämlich direkt im 1., indirekt im 2. Schnellabschaltkreis.

Direkter = 1. Schnellabschaltkreis: Die Auswahlschaltungen n(m) und die Verknüpfung zu einer Reihenschaltung werden bereits im o.g. Rangierverteiler realisiert. Diese Kontaktkette liegt symetrisch zwischen zwei Relais (-gruppen) mit kreuzweiser Selbsthaltung und Rückstellmöglichkeit. Die Schaltung arbeitet nach dem Ruhestrom-Prinzip; sie wird von einer potentialfreien Gleichspannung von 50V versorgt.

Indirekter = 2. Schnellabschaltkreis: Im Gegensatz zum 1. Kreis werden die Signale zunächst einer entsprechenden Anzahl von Relaispaaren zur nichtspeichernden/speichernden Kontaktvervielfachung zugeführt. Aus den Kontakten der nichtspeichernden Relais werden die Auswahlschaltungen zusammengestellt und aus ihnen eine Kontaktkette aufgebaut. Sie ergibt -analog zum 1. Kreis- zwischen zwei symetrischen Relais (-gruppen) mit kreuzweiser Selbsthaltung und Rückstellmöglichkeit den 2. Schnellabschaltkreis. Bis auf den Umstand der Signalverzögerung um eine Relaisentregungszeit ist der 2. funktionsidentisch mit dem 1. Kreis. Beide Kreise müssen unabhängig von einander die betreffenden Sicherheits-Funktionen auslösen; unterschiedliche Reaktionen lösen eine Störmeldung aus.

3. Zielsetzung für die elektronische Datenerfassung und -verarbeitung

a) Schnelle Datenerfassung aus dem Sicherheits-System: alle 1. und 2. Abschaltkreise, Absorbersteuerungen, nukleare und Brennelement-Instrumentierung.
b) Langsame Datenerfassung aus dem Sicherheits-System: alle übrigen Kreise der Logikschaltung, restliche Sicherheits-Instrumentierung.
c) Interpretation und Präsentation obiger Daten, ggf. kurz- und mittelfristige Archivierung.
d) Unterstützung, ggf. gesamte Ausführung von reaktortypischen Berechnungen: Abbrand, Flußverteilung, Strömungstechnik usw.
e) Umfangreiche Text-Datenbank für Prüf- und Bedienungsvorschriften, beschreibende Darstellungen, Terminverfolgung u.ä.

Aus Gründen der Verfügbarkeit sollte sich nur ein Minimum an datenerfassender und dokumentierender Peripherie des Rechners innerhalb der Reaktorhalle, dort aber unter Kontrolle des Schichtpersonals befinden. Damit war die Entfernung zu einem reaktorexternen Aufstellungsort für den Rechner mit mindestens 50m vorgegeben. Die datenerfassenden Komponenten sollten ferner unabhängig von firmengebundenen Prozeßelementen und mit Rücksicht auf die relativ inhomogene Datenpräsentation des Sicherheits-Systems des FRJ-2 weitgehend flexibel sein. Die Anforderungen an die Hard- und absehbare Software waren hauptsächlich bestimmt durch Umfang der schnellen Datenerfassung und -verarbeitung, nämlich min. 128 Binär-Signale/5msec und 32 Analogsignale/15msec zuzüglich einer zunächst nicht überschaubaren Anzahl von wesentlich langsameren Daten; dies entspricht einer Datenrate von ca. 4k/sec für das Gesamtproblem.

4. Betriebsrechner

Zur Realisierung der beschriebenen Vorstellungen wurde 1976 ein Rechner SIEMENS 330 mit Minimal-Peripherie angeschafft und zwischenzeitlich erweitert. Zur Realisierung von Pkt.3a) wurde für die Datenerfassung in der Reaktorwarte ein CAMAC-System installiert. In diesem dienen u.a. 4 Module mit je 2·16 statischen TTL-Eingängen der Verarbeitung der 128 schnellen binären, ein 32-kanaliger MPX mit nachgeschaltetem ADC (12bit/10V) der Verarbeitung der schnellen analogen Signale. Zu Überwachungszwecken ist jedes Crate mit einem Dataway-Display bestückt. Der alle Module eines Crate's verbindende Data-Highway mündet in den jeweiligen Crate-Controller. Letztere (derzeitiger Ausbau: 4 CC's) stehen miteinander über den Branch-Highway in Verbindung. Der Branch ist durch die Branch-Termination abgeschlossen, die auch ihrerseits ein Display enthält. Im Gegensatz zur Standardausführung eines Branch-Highway's konnte dieser wegen der Entfernung von 50m zum Rechner und der zu durchquerenden Reaktorhallenwand nicht auf übliche Weise an den Rechner angeschlossen werden. Vielmehr führt der Branch-Highway zunächst über einen 64bit Differential-Driver im Crate 1 und ein 128-poliges Kabel innerhalb der Reaktorhalle auf eine druckfeste Hallendurchführung mit Lötanschlüssen. Von der Außenseite der Durchführung verläuft das aufgetrennte Kabel weiter bis in den Rechnerraum. Hier sind ein weiterer Differential-Driver und der übliche System-Controller in einem eigenen Crate in der Nähe der ZE des Rechners untergebracht. Der SC liefert die Branch-Signale an eine Parallel-Schnittstelle im E/A-Prozessor der ZE. Die bisherigen Erfahrungen mit diesem sog. extended Branch sind positiv.

Bei der Installation war allerdings auf folgendes besonders zu achten: Die verdrillten Aderpaare des 128-poligen Kabels dürfen nicht versetzt verdrahtet werden - es entstehen sonst störende Echo-Signale bzw. unzulässiges Übersprechen zwischen den bit's. Es müssen eindeutige Erdungsverhältnisse zwischen den Stromversorgungen der ZE und der CAMAC-Peripherie herrschen - dies wurde durch eine rechnereigene E-Versorgung mit Schutzleitungs-System realisiert.

5. Datenkopplung: Sicherheits-System / Betriebsrechner

Wie alle sicherheitstechnisch relevanten Änderungen an der Reaktoranlage unterlag auch die Kopplung zwischen Sicherheits-System und Betriebsrechner dem üblichen Genehmigungs-Verfahren zwischen dem Anlagenbetreiber (KFA/ZFR) und der zuständigen Aufsichtsbehörde (Minister für Arbeit,Gesundheit und Soziales des Landes NRW); ferner wurde der Technische Überwachungsverein, Rheinland als Gutachter eingeschaltet.

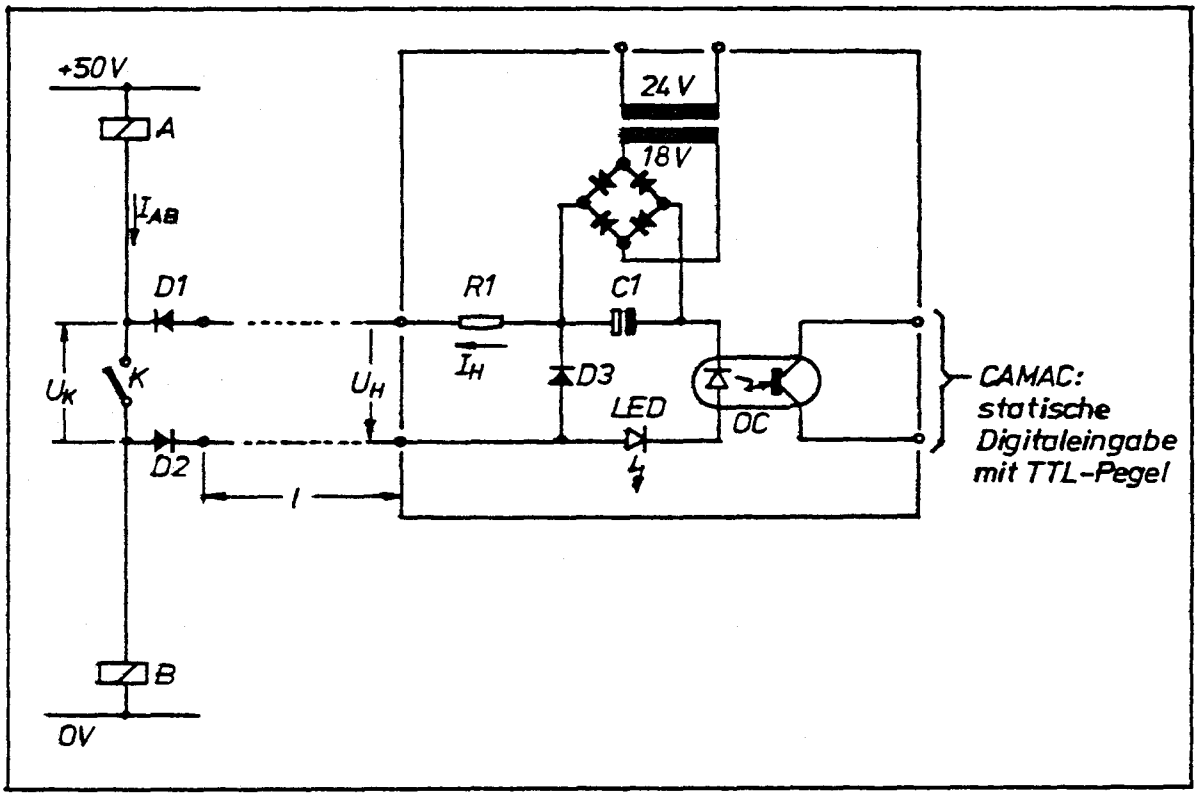
Aus der Vielfalt der zugelassenen und bisher z.T. realisierten Kopplungs-Möglichkeiten wird die der Binär-Daten im folgenden beispielhaft erläutert:

5.1 Binär-Daten

Für den Anschluß von 128 "schnellen" Binär-Signalen aus den 1. und 2. Abschaltkreisen und der Absorbersteuerung an 4 CAMAC-Module mit statischen Eingängen, sowie für 512 "langsame" Signale aus der restlichen Logikschaltung, deren Kopplung einem späteren Zeitpunkt vorbehalten bleiben soll, ließ sich auf Grund der DIDO-typischen Bauweise keine einheitliche "käufliche" Lösung finden. Daher wurden entsprechende Kopplungsverfahren bzw. Schaltungen in der ZFR selbst entworfen und hergestellt; diese haben neben ihrer spezifischen Aufgabe folgende Sicherheitsanforderungen zu erfüllen:

a) Es ist sicherzustellen, daß am Ort der Signalbereitstellung durch die zusätzliche Verdrahtung -ohne Rücksicht auf deren Ausdehnung- eine Überbrückung der Sicherheitsanregung ausgeschlossen wird.
b) Die Signale müssen noch innerhalb des primären Sicherheits-Bereiches (=Rangierverteiler-Raum) galvanisch getrennt und rückwirkungsfrei zur Verfügung stehen; die Rückwirkungsfreiheit ist redundant auszulegen.
c) Die Entkopplung muß für eine Prüfspannung von 1kV ausgelegt sein.
d) Die für die Entkopplung erforderlichen Hilfsspannungen dürfen sich nicht zu unzulässig hohen Berührungsspannungen addieren.

Das folgende Bild zeigt die Lösung am Beispiel der Überwachung eines Grenzwertkontaktes K: An der betreffenden Übergabeleiste im Rangierverteilerraum wird über 2 Dioden D1 und D2 unterschiedlichen Typs mit 1kV Sperrspannung ausgekoppelt. Sie halten bei geöffnetem Kontakt die Betriebsspannung U_K=50V von der nachfolgenden Schaltung fern und verhindern außerdem, daß ein beliebiger Schluß in der Leitungsführung l die Wirkung von K auf die Relais A und B des Abschaltkreises aufhebt. Etwa 10m entfernt von der "Anzapf-Stelle" befindet sich die Trennschaltung. Bei geschlossenem Kontakt K treibt die Hilfsspannung an C1 einen Strom über R1, der ausreicht, um den Optokoppler OC voll auszusteuern. Die LED dient hierbei als Funktionskontrolle;



D3 hält induktive Spannungsspitzen aus der Relaisschaltung vom OC fern. R1 ist so bemessen, daß auch bei Schluß aller Dioden die Trennschaltung allein die Wirkung von K nicht aufheben kann. Im Prinzip wird jedes Anrege-Kriterium in den eingangs genannten Abschalt- und Steuerkreisen auf diese Weise einzeln entkoppelt. Je 3 Trennschaltungen sind auf einer Platine untergebracht; 128 Trennschaltungen, zuzüglich Reserve-Einheiten befinden sich in einem 19"-Rahmen.

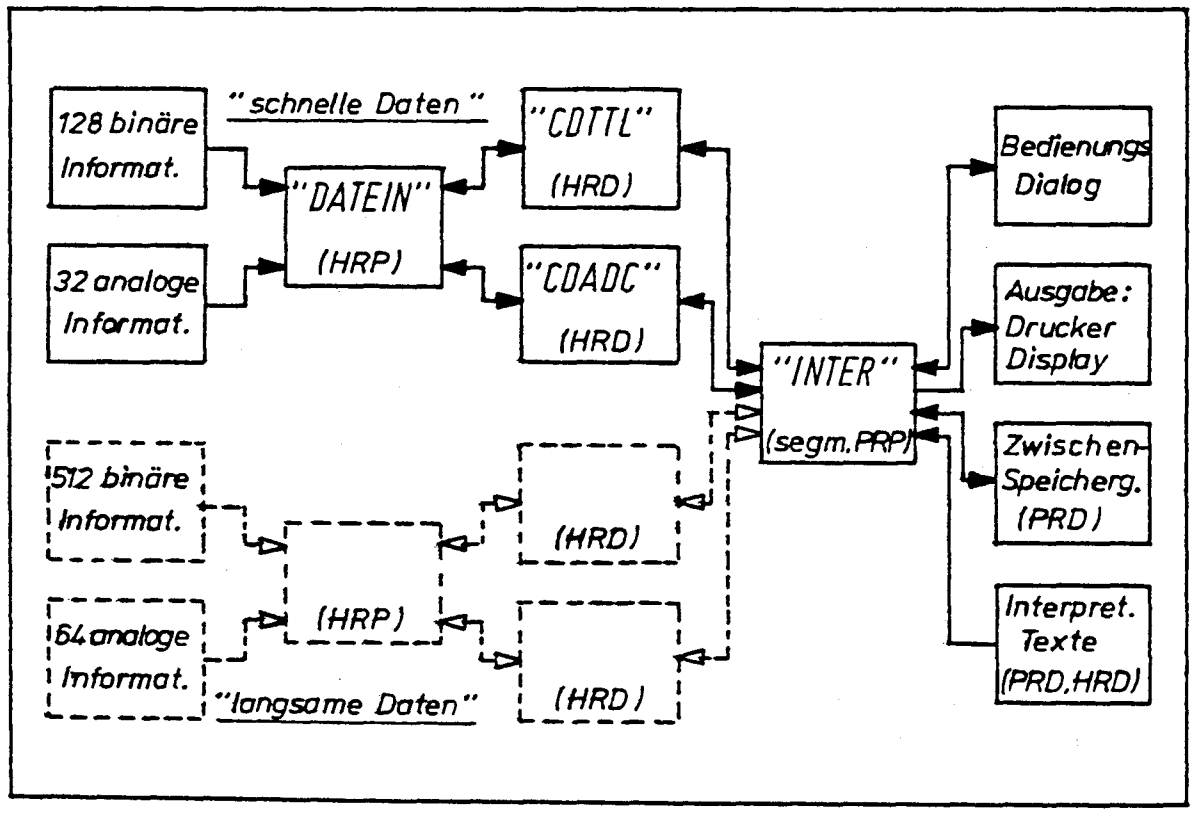
6. Reaktorbetrieb

Bedingt durch den Abbrand des U-235 mit etwa 1,25 g/MWd, aber auch mit Rücksicht auf experimentelle Belange ergibt sich ein diskontinuierlicher Reaktorbetrieb der Art, daß auf ca. 3 Wochen Betrieb mit Nennleistung eine Abschaltpause folgt. Obgleich stets ein unterbrechungsfreier 3-Wochen-Betrieb angestrebt wird, sind Störungen durch automatische Abschaltungen des Reaktors nicht immer zu vermeiden. Sie behindern die experimentelle Nutzung um so mehr, da der Reaktor nach Unterbrechung des Leistungsbetriebes je nach Kernzustand binnen 15...30min erneut auf Nennleistung angefahren sein muß. Dieser Umstand basiert auf der Vergiftung des Brennstoffes mit dem Sekundär-Spaltprodukt Xe-135, dessen Aufbau über der Betriebsgleichgewichtsvergiftung sofort nach Reaktorabschaltung beginnt. Der Abbau der Vergiftung auf ein wiederanfahrbares Maß dauert 36...45 Stunden. Man ist daher nach automatischen Reaktorabschaltungen bestrebt, Störungserkennung, Gegenmaßnahmen, Prüfungen und Wiederanfahren in der o.g. Zeit zu schaffen. Dies war vor Einführung der "schnellen Datenerfassung und -verarbeitung" mit den zur Verfügung stehenden konventionellen Mitteln der Meß-, Melde- und Registriertechnik nicht immer möglich.

7. Software

Die im folgenden zu erläuternde Software hat nicht so sehr die Aufgabe die traditionellen Verfahren des Reaktor-Sicherheits-

Systems zu ersetzen, als vielmehr Informationslücken bei der Störungsursache dort zu schließen, wo Datendichte nach Anzahl und Zeit bisher keine Auflösung mehr zuließ. Außerdem sollen vorbeugende Wartung und Reparatur durch Trendverfolgung unterstützt werden. Die von SIEMENS z.V. gestellte CAMAC-Software erwies sich zwar als universell und leicht einsetzbar, aber vor allem im Hinblick auf ihre ORG-Abhängigkeit für die genannten Zwecke als zu zeitaufwendig. Für die Belange der Datenverarbeitung am FRJ-2 hat sich die Entwicklung von Software, die genau auf die Möglichkeiten und Erfordernisse des DIDO-Sicherheits-Systems zugeschnitten ist, als deutlich wirkungsvoller herausgestellt. Da vor allem auf eine schnelle Erfassung einer Teilmenge der Gesamtinfomationen aus dem Sicherheits-System Wert gelegt wurde, ergab sich zwangsläufig eine Software-Struktur, bei der die Phasen Erfassung / Vorverarbeitung und Interpretation / Weiterverarbeitung streng getrennt sind. Als informative Schnittstellen zwischen den beiden Phasen dienen Common-Data-Bereiche. Das folgende Bild zeigt die Gesamtstruktur:



Die "schnellen Daten", bestehend aus 128 binären und 32 analogen Informationen werden vom HRP "DATEIN" aus dem CAMAC-System erfaßt und vorverarbeitet; Ergebnisse werden ggf. in den CD's CDTTL und CDADC hinterlegt. Sinngemäß werden die als "langsame Daten" bezeichneten 512 binären und 64 analogen Informationen behandelt. Das segmentierte PRP "INTER" hat vielfältige, aber i.W. zeitunkritische Aufgaben zu erfüllen. Von ihm geht vor allem der koordinierende Einsatz der HRP anhand des Bedienungs-Dialoges aus. INTER hat ferner den Inhalt der HRD zu interpretieren und sofort auszugeben, oder als PRD zwischenzuspeichern. Letzterenfalls findet die Ausgabe bedarfsweise später statt. Zur Kennzeichnung der binären und analogen Informationen stehen vorformulierte Interpretationstexte größtenteils als PRD, aber auch als CD zur Verfügung. Die "langsame Datenerfassung und -verarbeitung" erfolgt entweder

im festen Zeittakt oder nach Maßgabe der Ereignisse im Bereich der "schnellen Daten". Die HRP und HRD sind so strukturiert, daß sie außer von INTER auch von anderen Programmen angesprochen werden können. Im Rahmen der Software-Entwicklung wurde besonderer Wert auf einen Bedienungs-Dialog gelegt, der es gestattet, auch Rechner-unorientiertes bzw. unroutiniertes Personal mit der Handhabung der Programme zu betrauen.

7.1 Programm "DATEIN"

Das Programm DATEIN dient der Abfrage, Vorverarbeitung und ggf. Abspeicherung von Daten aus dem CAMAC-System nach Maßgabe eines Moduswortes. Es wird vom übergeordneten Programm INTER, welches danach selbst auf Fortsetzung wartet, gestartet. Abgesehen von Anfangs- und Endroutinen läuft DATEIN in Ebene 2.0; es kann daher mit ORG- bzw. Mitteln der Standard-Bedienung nicht unterbrochen werden. Weil DATEIN bei der Vorverarbeitung der aufgenommenen Daten hauptsächlich auf das Erkennen von Änderungen und nur ausnahmsweise auf die Weitergabe statischer Information hin ausgelegt ist, führt nur ein vordefiniertes Ereignis aus diesem Programm regulär wieder in das auf Fortsetzung wartende zurück.

DATEIN bearbeitet je Start wahlweise die Gruppe der 128 binären oder die der 32 analogen Signale. Kumulierung ist wegen der allzu verschiedenen Auflösung der beiden Gruppen, aber auch wegen der kausalen Abhängigkeit nicht sinnvoll. Ferner ist die Verarbeitungsgeschwindigkeit nach "fast mode" und "slow mode" zu unterscheiden. Der fast mode ist nur abhängig von der Summe aller hardwarebedingten Operationszeiten des durchlaufenen Programmteils; die software ist daraufhin optimiert. Der slow mode entsteht durch Einschieben von 2 Totzeitschleifen an geeigneter Stelle des Programms DATEIN. Die Gesamttotzeit ist mittels zweier Faktoren F1 und F2 in weiten Bereichen definierbar. Zusätzlich kann das Programm DATEIN für zyklischen oder einmaligen Durchlauf präpariert werden. Alle genannten Parameter werden über die CD-Bereiche zur Verfügung gestellt.

7.1.1. Binär-Teil

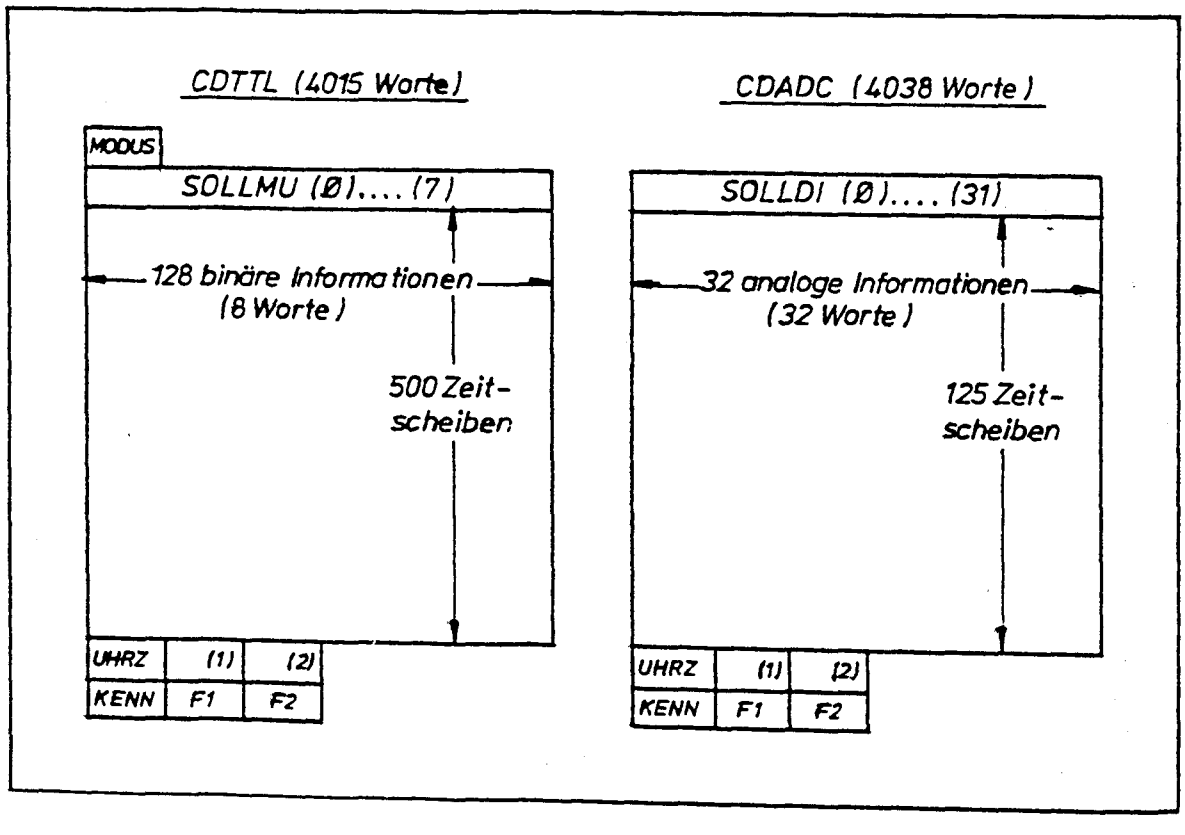
128 binäre Signale werden in einer zeitlich möglichst dichten Sequenz von 8 Datenworten aus dem CAMAC-System in einen Istwertpuffer übertragen und anschließend mit einem ebenfalls 8 Worte langen binären Sollmuster verglichen. Die zwischen der Erfassung des 1. und 8. Datenwortes bestehende Zeitdifferenz wird dabei vernachlässigt. Man betrachtet den Inhalt des Istwert-Puffers als zeitidentisches Abbild der überwachten Relaislogik-Schaltung des Sicherheits-Systems in Form einer "Zeitscheibe". In Abhängigkeit vom fast/slow mode bzw. den Faktoren F1, F2 werden nun in festen Abständen solche Zeitscheiben zyklisch bearbeitet. Es ist wegen der zusätzlichen logischen Abfragen nicht effizient, variable Untermengen der 128 Informationen zu bearbeiten; partielle Weiterverarbeitung bleibt dem Programm INTER vorbehalten. Das benötigte Sollmuster wird dem entsprechenden CD-Bereich entnommen. Wird eine Soll-/Istdifferenz festgestellt, so liest DATEIN ohne weiteren Vergleich, aber mit der gleichen Taktfrequenz 500 Zeitscheiben in den zugehörigen CD-Bereich ein; dieses Datenfeld präsentiert dann, beginnend mit mindestens einer Abweichung vom ggf. stationären Zustand, ein 128 bit breites und 500 Zeitscheiben langes Verhaltensprofil der betreffenden Relaislogik des Sicherheits-Systems.

Danach wird die aktuelle Uhrzeit in den CD-Bereich eingetragen, das 8-Worte-Istmuster als zukünftiges Sollmuster zurückgeschrieben und das wartende, übergeordnete Programm INTER fortgesetzt. Im fast mode beträgt der Zeitscheibenabstand 333µsec. Unter Berücksichtigung der kritischen Relaiszeit der Logikschaltung von 5msec reicht die Überwachungsmöglichkeit damit vom Aufspüren von Kontaktprellen, extremen Wackelkontakten und ähnlichen Störungen mit oberer Grenzfrequenz von 3kHz bis zu langfristigen Dokumentationsaufgaben bei Anwendung des slow mode. In der Praxis haben sich Standardeinstellungen von 2...5 msec bei kontinuierlicher Überwachung (=Reaktorleistungs-Betrieb) bewährt.

7.1.2. Analog-Teil

Für die in diesem Teil des Programms DATEIN zu verarbeitenden 32 analogen Signale gilt vieles sinngemäß zum vorangegangenen Abschnitt. Die Daten werden jedoch nicht gegen Absolutwerte, sondern gegen Differenzen/Zeitscheibe verglichen. Das resultierende Datenfeld präsentiert hier, mit mindestens einer signifikanten Differenzüberschreitung in der 2. Zeitscheibe gegenüber der 1., ein 32 Analogwerte breites und 125 Zeitscheiben langes Signal-Profil der betreffenden Instrumentierung des Sicherheits-Systems. Im fast mode beträgt der Zeitscheibenabstand 5,25msec. Unter Berücksichtigung der kritischen Signallaufzeit der Instrumentierung von 15msec reicht die Überwachungsmöglichkeit damit von Störungen aller Art mit einer oberen Grenzfrequenz von 200Hz bis zu langfristigen Dokumentationsaufgaben bei Anwendung des slow mode. In der Praxis haben sich Standardeinstellungen von 10...15msec bewährt.

7.2. CD-Bereiche



7.3. Programm "INTER"

Die Aufgaben dieses Programms sind recht komplex; ständige Anpassung bzw. Erweiterung im Hinblick auf geänderte Reaktorbetriebsweise, akute Störungsprobleme, aber auch auf Grund der laufenden Erfahrungen mit der "schnellen Datenerfassung und -verarbeitung" ist erforderlich. Im folgenden wird daher nur der Funktionsbereich vorgestellt, der unmittelbar zum Thema gehört:

a) Koordinierung aller für die Datenerfassung mit CAMAC tätigen HRP, z.Bsp. DATEIN, anhand von Bedienungseingaben oder logischen Entscheidungen aufgrund von vordefinierten Daten-Konfigurationen.
b) Bedienungs-Dialog mit "geführter Bedienung"
c) Sofort-Interpretation der im betreffenden CD-Bereich übergebenen Daten und Listenausgabe im zyklischen Betrieb, oder:
d) Zwischenspeicherung der betreffenden CD's auf Wechselplatte.
e) Nach Beendigung des zyklischen Betriebes: Such-Routinen und Informationen ähnlich c).
f) Einzeloperationen bestehend aus Datenerfassung und gezielter -präsentation anhand von Bedienungseingaben.

Unter Interpretation sind im Rahmen des Programms INTER je nach Reaktionsart binär / analog folgende Arbeitsschritte zu verstehen:

a) Auswertung des Moduswortes
b) Bit-weise Zustandserkennung je Zeitscheibe im CDTTL, $^{3}/_{4}$-Wort-weise Differenzerkennung je Zeitscheibe im CDADC; Absolutierung.
c) Zeitliche Zuordnung der Zeitscheiben mittels UHRZ, KENN, F1, F2 des betreffenden CD's.
d) Zuordnung Bit-Nr / Kanal-Nr. ---> Interpretationstext.
e) Ausgabe der gesamten Interpretation in verschiedenen Darbietungsformen.

INTER enthält grundsätzlich keine zeitkritischen Funktionen. Man darf jedoch nicht übersehen, daß im zyklischen Betrieb INTER <---> DATEIN (Fortsetzstart <---> Warten auf Fortsetzung) keine Überwachung des Sicherheits-Systems seitens CAMAC erfolgt, solange INTER aktiv ist. Betroffene Programmteile sind daher so "sparsam" wie möglich ausgestattet. Die bisherige Betriebserfahrung zeigt, daß die auf diese Weise verlorengegangenen Ereignisse zahlenmäßig kaum ins Gewicht fallen und meist auch nicht sehr aufschlußreich sind, da sie ja zeitlich stets nach einer störungsbedingten Datenverarbeitung auftreten.

Belegen und Freigeben für ORG K

E. Bachner, Kernforschungsanlage Jülich/ZAT

Zu Anfang soll für diejenigen, die das ORG PP II nicht kennen, kurz erklärt werden, was Belegen und Freigeben bedeutet.

In einem Prozeßrechner laufen normalerweise mehrere Programme simultan zueinander ab. Je nach Aufgabenstellung benutzen sie dieselben Dateien und/oder Externgeräte (z.B. ein Protokoll-Gerät). Da Protokollierprogramme selten zeitlich koordiniert ablaufen, können sie durchaus auch gleichzeitig ein Externgerät benutzen wollen. Falls dieses Gerät z.B. ein Drucker ist, gibt es beim ORG K Protokoll-Salat.
Ein Gerät belegen bedeutet nun, daß ein Programm ein Gerät oder eine Datei solange zugeteilt bekommt, bis es von diesem Programm wieder freigegeben wird, bzw. bis das Programm beendet wird.
Will ein anderes Programm zur gleichen Zeit dieses Gerät benutzen dann wird es solange wartend gesetzt, bis das Gerät wieder frei ist.
Das Belegen und Freigeben ist im Grunde nichts weiter als eine Koordinierungsfunktion, die auch mit dem Koordinierungszähler zu realisieren wäre. Das setzt aber Vereinbarungen über Koordinierungszählernamen voraus. Werden nur eigene Programme eingesetzt, ist diese Methode zwar aufwendiger, aber durchaus brauchbar. Werden zusätzlich auch Dienstprogramme von SIEMENS verwendet (AS3O,FC3O), funktioniert es nicht mehr, da die SIEMENS-Programme diese Koordinierungsmöglichkeit nicht kennen. Der Aufwand dafür schien SIEMENS jedenfalls größer, als die Belegfunktion in das ORG K zu integrieren.

Belegfunktion

Die Belegfunktion besteht aus zwei Teilen:

1. dem ORG-Zusatz BELGCD, der die eigentliche Funktion ausführt, und
2. dem Bedienungsprogramm BELGBE mit den Aufgaben:
   - Installation des Systems
   - Aufbau der notwendigen Listen
   - Bedienung des Systems.

Vergleicht man unsere Version der Belegfunktion mit der des ORG PP II, so ergeben sich einige Einschränkungen, aber auch einige Verbesserungen.

Einschränkungen:

1. Es gibt nur noch einen Belegmodus:
   exklusiv
   gegenüber drei beim ORG PP II.
2. Es können keine Dateien und keine Eingabe-Geräte belegt werden.
3. Es können nur bestimmte Ausgabe-Geräte belegt werden.

Diese Einschränkungen scheinen uns nicht gravierend zu sein, da sie entweder durch die Koordinierungszählerfunktion ersetzt werden können (z.B. bei Dateien) oder praktisch nie gebraucht werden (Eingabe-Geräte, mehrere Belegmodi).

Vorteile:

Beim Besprechen der Vorteile unserer Version muß auch gleich auf die Bedienungsanweisungen von BELGBE eingegangen werden. Die Belegfunktion wird nicht durch ORG-Generierung installiert, sondern sie kann im Prinzip zu jedem beliebigen Zeitpunkt in ein laufendes System eingebracht werden. Einzige Voraussetzung ist ein freier Speicherbereich von ca. 300 Worten im Anschluß an das ORG. Dorthin wird der ORG-Teil BELGCD geladen.

Mit der Bedienung

GER: 1..10 { ger } ;

werden die Geräte, die der Belegfunktion unterworfen werden sollen, in eine Liste eingetragen. Diese Geräte-Liste kann zu jedem beliebigen Zeitpunkt erweitert werden. Allerdings ist es nicht möglich, diese Zuordnung auf einfache Weise wieder rückgängig zu machen. Das ist nur möglich durch Ausschalten der Belegfunktion mit "AUS" und einer Neuinstallation. Der Aufwand für eine elegantere Lösung schien uns nicht gerechtfertigt.
Wird die Belegfunktion jetzt mit

EIN;

aktiviert, dann wird das explizite Belegen wirksam, d.h. ein Gerät wird nur mit einem ØBELG belegt. Diese Art des Belegens gibt es im ORG PPII nicht. Dort wird in jedem Fall beim ersten E/A-Aufruf belegt. Dieses implizite Belegen gibt es aber ebenfalls in einer erweiterten Form. Mit der Anweisung

PNR: 1..10 { bereich } ;

können Programme eingetragen werden, die auch implizit belegen. Werden alle Programme angegeben, ergeben sich dieselben Verhältnisse wie beim ORG PPII.
Es kann durchaus sinnvoll sein, Programme vom Belegmechanismus auszunehmen. Es gibt z.B. Prozessprogramme, die nur selten eine Meldung ausgeben, aber nie beendet werden, sondern über einen Koordinierungszähler gesteuert werden. Sie müßten das Ausgabegerät immer explizit freigeben, um es nicht zu blockieren.
Das ist nicht notwendig, wenn das Programm nicht eingetragen wird und damit als nicht belegpflichtig erklärt wird.
Die Liste der belegpflichtigen Programme kann wieder zu jedem beliebigen Zeitpunkt erweitert aber mit der Anweisung

LOE: 1..10 { bereich } ;

auch verkleinert werden;

Freigabe:

Ein belegtes Gerät wird freigegeben

1. nach einem Aufruf $FREI,
2. am Ende eines Programms
3. bei einem $BEDIEN mit Warten
4. bei einem Abbruch des Programms durch das ORG

Bei den Warte-Aufrufen $KOOR und $WARTFO werden die Geräte nicht freigegeben. Tritt dadurch eine Blockierung eines Gerätes auf, dann besteht mit der Bedienung

FREI:<ger>;

die Möglichkeit, dieses Gerät wieder freizugeben. Beim ORG PPII mußte das fehlerhaft belegende Programm mit /ENDE zwangsbeendet werden, um die Blockierung aufzuheben. Da das ORG K am Programmende im Gegensatz zum ORG PPII aber keine Warteschlangen-Austräge vornimmt, besteht diese Möglichkeit hier nicht.

Mit der Bedienungsanweisung

PROT;

kann die Geräte- und Programmnummern-Liste protokolliert werden.

Der ORG-Zusatz BELGCD und sämtliche Listen werden auch in das ORG-Abbild eingetragen, sodaß nach einem Wiederanlauf die Belegfunktion wieder in dem Zustand ist wie vor dem Wiederanlauf.

Probleme bei BELG/FREI
==========

Einige Programme machen trotz der Belegfunktion Kummer. Dazu gehören FORTRAN-Programme und der Monitor 300.

FORTRAN:
Mit einem FORTRAN-Programm ist es selbst mit der Belegfunktion nicht möglich, ein zusammenhängendes Protrokoll zu erhalten. Ein mit dem FC30 übersetztes FORTRAN-Programm gibt das Ausgabegerät bei einer WRITE-Anweisung nach jedem FORTRAN-Satz, d.h. nach jeder Zeile explizit frei. Siemens war trotz Bitten

zu einer Änderung des FC30 nicht bereit. Die Änderung könnte darin bestehen, entweder das explizite Freigeben ganz herauszunehmen, oder die Freigabe erst am Ende einer WRITE-Anweisung auszuführen.

MONITOR:
Ein ähnliches Problem tritt beim MONI30 auf. Er gibt sein Protokollgerät ebenfalls nach jeder Zeile explizit frei.
Gibt ein anderes Programm während eines Monitorlaufes ein Protokoll aus, so ist das Protokoll jetzt zwar zusammenhängend, aber da ja nach einem Protokoll normalerweise kein Blattvorschub gemacht wird, schreibt der Monitor auf dem Protokollblatt seine Meldungen weiter.
Um auch diesen Problemen zu begegnen, haben wir eine weitere Version des ORG-Zusatzes BELGCD gemacht. Die wichtigste Änderung besteht darin, daß der Aufruf $FREI nicht mehr ausgewertet wird. Damit sind die Probleme mit den FORTRAN Programmen behoben. Der MONITOR belegt jetzt während des gesamten Monitorlaufes das Protokoll-Gerät. Damit aber Compiler und Übersetzer trotzdem auf demselben Gerät ihr Protokoll ausgeben können, wird das MAP-Bit der PNULI abgefragt. Bei Programmen, die dem Monitor unterstellt sind (MAP's), wird dann die Belegung des Protokollgerätes durchbrochen.
Da der Aufruf $FREI jetzt für alle anderen Programme auch keine Wirkung mehr hat, müssen eventuelle Folgen beim Einsatz dieser Version bedacht werden.

Ein experimentorientiertes Bereichsrechenzentrum in der KFA Jülich

H. Huppertz, W. Janßen, J. Lerch,
P. Reinhart, F. Rongen, W. Tenten

Zentrallabor für Elektronik
der Kernforschungsanlage Jülich GmbH

Ziel dieses Berichtes über ein experimentorientiertes Bereichsrechenzentrum in der Kernforschungsanlage Jülich (KFA) ist es, die Siemens 306-Installation des Zentrallabors für Elektronik (ZEL) darzustellen, wie sie sich im Laufe der siebziger Jahre entwickelt hat.

Über einen Teil der Komponenten wurde bereits auf früheren SAK-Tagungen berichtet /1/.

In der Abb. 1 ist eine Übersicht über die Installation des 306-Rechners in der KFA gezeigt. Den Kern bildet eine auf 96 k Worte ausgebaute Zentraleinheit vom Typ Siemens 306. Der Grund für diesen für einen 306-Rechner ungewöhnlich hohen Speicherausbau ist darin zu suchen, daß ein Teil der von Siemens gelieferten Kernspeicherblöcke durch vom ZEL entwickelte Halbleiterspeicher ersetzt wurde, wie es in Abb. 2 gezeigt ist.

Dadurch alleine ist es aber noch nicht möglich, über 64 k hinauszukommen. Von den Entwicklern der 306 war eine Umschaltmöglichkeit vorgesehen, um anstelle der 16k-Module 32k-Module einzusetzen.

Wenn man den in Abb. 2 angedeuteten Schalter in der Weise benutzt, wie es von den Entwicklern der 306 geplant wurde, ist ein Ausbau auf 128 k möglich.

Als Nachteil bei dieser Lösung (Abb. 3) ergibt sich aber die Einschränkung, daß man keinen Original-Kernspeicherblock mehr verwenden kann, und daß der Ausbau nur in 32k-Stufen vorgenommen werden kann. Daher wurde eine Zwischenlösung gefunden und verwirklicht, wie sie in Abb. 4 gezeigt ist.

Durch geringfügige Modifikationen in der Adress-Dekodierung im Rechen-Steuerwerk kann man jetzt Kernspeicher und Halbleiterspeicher nach Bedarf gemischt verwenden. Im ZEL tatsächlich installiert sind 2 Kernspeicherblöcke, an Nahtstelle 3 ein Halbleiterspeicher und an der sozusagen aufgebohrten Nahtstelle 4 3 Halbleiterspeicher. Prinzipiell kann man bei Bedarf noch zwei Blöcke hinzufügen und damit den Maximalausbau erreichen. Für alle diese Erweiterungen sind im Betriebssystem keinerlei Änderungen erforderlich.

Für den Anschluß von Experimentrechnern, die über das KFA-Gelände verteilt sind, an die 306 wurde vor etwa 10 Jahren im ZEL eine Rechnerkopplungseinheit /2/ entwickelt, die an einen Standardkanal der 306 angeschlossen wurde (Abb. 1). Diese Kopplung war für die zu überbrückenden Entfernungen recht leistungsfähig. Wenn man die Leistungsfähigkeit einer Rechnerkopplung aber nur durch die reine Übertragungsgeschwindigkeit beschreibt, so ist das leicht irreführend. Zusätzlich zur reinen Übertragungszeit muß man noch andere Zeiten berücksichtigen, die man braucht, um zum Beispiel eine Verbindung erst einmal aufzubauen, einen Dialog in den verschiedenen Phasen der Übertragung zu führen oder die übertragenen Daten anschließend zu verarbeiten. Im Vergleich zu all diesen Zeiten ist die reine Übertragungszeit weniger wichtig.

Diese Rechnerkopplungseinheit hatte auch einige Nachteile, die einer Verwendung in größerem Umfange entgegenstanden. Da war zum Beispiel der Leitungsbedarf von 14 Paaren pro Experimentrechner. An ein Kopplungselement konnten maximal 6 Experimentrechner angeschlossen werden. Hier zeigte sich für den weiteren Ausbau ein Engpaß ab.

Ein ähnlicher Engpaß bestand in der Verwendung des Fernschreibelementes FSK für den Anschluß von graphischen Sichtgeräten an den 306-Rechner. Diese Sichtgeräte werden für die interaktive Auswertung von Experimentdaten benötigt /3/. Obwohl Probleme, wie die Verwendung des ASCII-Codes anstelle des CCIT-Fernschreib-Codes, erfolgreich gelöst wurden, war die Verwendung des Fernschreibelementes für den Anschluß einer Vielzahl von Terminals nicht die günstigste Lösung.

Aus diesen Gründen wurde eine Alternative gesucht und auch gefunden, die für KFA-Verhältnisse keinerlei Einschränkungen hinsichtlich möglicher Erweiterungen mit sich brachte. Die Lösung lag in der Verwendung von CAMAC-Modulen für den Anschluß von Experiment-Rechnern und von Sichtgeräten. Da die 306 nicht über einen kommerziell erhältlichen CAMAC-Anschluß verfügte und auch nicht durch zu viele triviale Aufgaben belastet werden sollte, wurde ein Siemens-320-Rechner als Organisationsrechner zwischengeschaltet, der über eine vom ZEL entwickelte schnelle Kopplung an einen Schnellkanal der 306 angeschlossen wurde /4/. Übrigens wurden auch für den 320-Rechner preiswerte Halbleiter-Speichermodule entwickelt /5/.

Im Augenblick sind über die 320 an die 306 6 graphische Sichtgeräte vom Typ Tektronix 4010 bzw. 4012 angeschlossen. Als alphanumerische Terminals können auch Sichtgeräte der hp-Serie 2640 oder Teletypes verwendet werden. Im Laufe der Zeit sind auch 3 Vielkanalanalysatoren dazugekommen, die ihre Daten auf diesem Wege in das Zentrallabor für Elektronik übertragen. Neu hinzugekommen sind zwei Tischrechner der Tektronix 4050-Serie, die als Experimentrechner zur Datenerfassung und Steuerung dienen. Gleichzeitig können diese Rechner aber auch als Terminal arbeiten. Da die Plotter Tektronix 4662 weitgehend formatkompatibel zu den 4010-Terminals sind, können sie auch problemlos betrieben werden. Alle diese Geräte stehen im KFA-Gelände und sind über jeweils 2 Paare mit Telefon-Standleitungen voll duplex angeschlossen.

Die neueste Anwendung ist der Anschluß eines aus ca. 70 Stationen bestehenden Datenerfassungssystems für Solar-Energie-Daten. Die ersten 4 Stationen, die im Augenblick aufgebaut sind, stehen in der KFA, in Unna, auf Borkum und in Achern. Die mit einem Mikro-Computer ausgerüsteten Stationen sammeln laufend Wetterdaten und Daten von Brauchwasseranlagen. Zu festgelegten Zeiten werden zweimal am Tage alle Stationen nacheinander von der 306 angewählt, um die bis dahin gemessenen Datenblöcke einzusammeln. Die Anwahl und die Datenübertragung erfolgt mit Hilfe eines Modems und einer automatischen Wähleinrichtung über das öffentliche Telefonnetz der Deutschen Bundespost. Die Telefonnummern der Stationen sind dabei in der 306 hinterlegt. Erst wenn sichergestellt ist, daß alle übertragenen Meßwerte auf der Datenplatte der 306 abgespeichert sind, werden sie in den Meßstationen gelöscht.

Durch diese vielfältigen Anforderungen auf dem On-Line- und Timesharing-Sektor, zu denen sich noch einige zeitunkritische Stapelverarbeitungsaufgaben hinzuaddieren, wäre es früher oder später zu Speicher-Engpässen in der 306 gekommen. Der Mangel an mehreren genügend großen Laufbereichen und damit an Arbeitsspeicher wurde, wie oben angezeigt, zufriedenstellend gelöst. Ein weiterer Engpaß besteht im Augenblick noch in der Plattenspeicherkapazität. Insbesondere wegen der Vielzahl der anfallenden Daten, die für längere Zeit on-line zur Verfügung gestellt werden müssen, sind die beiden Siemens-Platten vom Typ PS 22 viel zu klein.

Eine Erweiterung unter Verwendung dieser Laufwerke ist aber weder möglich, noch erstrebenswert wegen der Zahl der Schnellkanäle und dem von Siemens angewandten Steuerungsprinzip. An einem von nur 5 vorhandenen Schnellkanälen der 306 kann über ein als Grundsteuerung arbeitendes Prozeßelement P3K immer nur ein Laufwerk angeschlossen werden. Diese unsinnige Einschränkung wurde als erste Maßnahme aufgehoben, indem über einen kleinen selbstgebauten Multiplexer beide PS 22-Laufwerke an ein einziges P3K angeschlossen wurden (Abb. 5). Die Umschaltung des Multiplexers geschieht über ein Bit in dem bis dahin ungenützten Maskenkanal des P3K.

Damit war das zweite P3K mit der zugehörigen Schnellkanal-nahtstelle frei. Hierzu passend wurde inzwischen eine neue Plattensteuerung entwickelt und gebaut, an die bis zu 4 Laufwerke modernster Art angeschlossen werden können. Zum Einsatz kommt hier die Type 9766 der Firma CDC, die auch im System-300-16-Bit als Plattenspeicher 3948 eingesetzt wird. Diese Laufwerke entsprechen auch vom Wartungsaufwand und von der Zuverlässigkeit her dem Stand der Technik; sie besitzen zum Beispiel keine Hydraulik mehr. Ein CDC-Laufwerk hat dabei die Kapazität von mehr als 22 PS 22-Laufwerken.

Dem Betriebssystem gegenüber verhält sich eine neue Platte wie 22 Einzelplatten. Für die Umschaltung zwischen den Teil-Platten wird ebenfalls das Maskenregister des P3K benutzt. So konnten die Änderungen im Betriebssystem minimal gehalten werden.

Im Augenblick befindet sich die Hardware der Plattensteuerung zwar noch in der Erprobung, die entsprechende Software-Änderung im Betriebssystem ist seit einiger Zeit schon in Betrieb, da sie identisch ist mit der für die Umschaltung der beiden über ein einziges P3K angeschlossenen PS 22-Laufwerke.

/1/ H. Huppertz, P. Reinhart, F. Rongen, W. Tenten:
Speichererweiterung für die Siemens 306,
Tagungsbericht der 9. Jahrestagung des Siemens Prozeßrechner-Anwenderkreises I vom 5. bis 7. April 1978 im Kernforschungszentrum Karlsruhe, KfK 2642 (1978), p. 272 - 279.

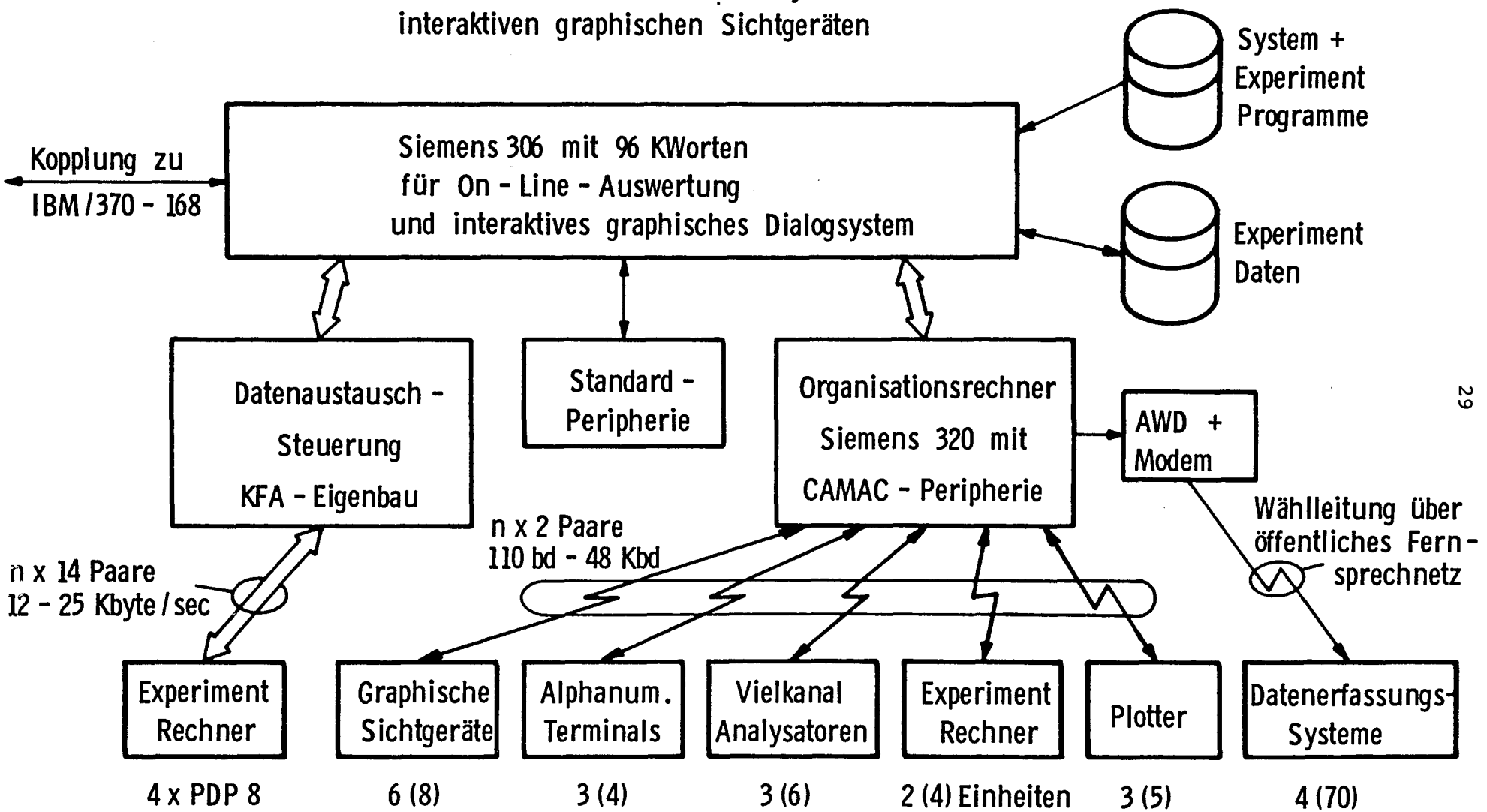
/2/ F. Rongen:
Kopplung von verschiedenen Experimentrechnern mit einer Siemens 306/320-Installation über CAMAC,
Tagungsbericht der 5. Jahrestagung des Anwenderkreises I Siemens Prozeßrechner vom 2. bis 4. April 1974 in der Physikalisch-Technischen Bundesanstalt Braunschweig,
PTB-FMRB-54(1974), p. 197 - 207.

/3/ J. Lerch:
DIAL - Ein Display-Dialog-Programmsystem zur Experimentdatenauswertung für die Siemens 305-306,
Tagungsbericht der 5. Jahrestagung des Anwenderkreises I Siemens Prozeßrechner vom 2. bis 4. April 1974 in der Physikalisch-Technischen Bundesanstalt Braunschweig,
PTB-FMRB-54(1974), p. 109 - 124

/4/ H. Huppertz, W. Janßen, P. Reinhart, F. Rongen, W. Tenten:
Mikroprogrammiertes CAMAC-Rechnerkopplungsmodul,
Tagungsbericht der 7. Jahrestagung des Anwenderkreises I Siemens Prozeßrechner vom 21. - 23. April 1976 in der Fachhochschule Ulm, p. 127 - 180

/5/ H. Huppertz, P. Reinhart, W. Tenten:
Ein 8 k Halbleiterspeichermodul für das Siemens-System 300-16 Bit,
Tagungsbericht der 8. Jahrestagung des Anwenderkreises I Siemens Prozeßrechner vom 20. - 22. April 1977 in der Fachhochschule Dortmund, p. 128 - 131

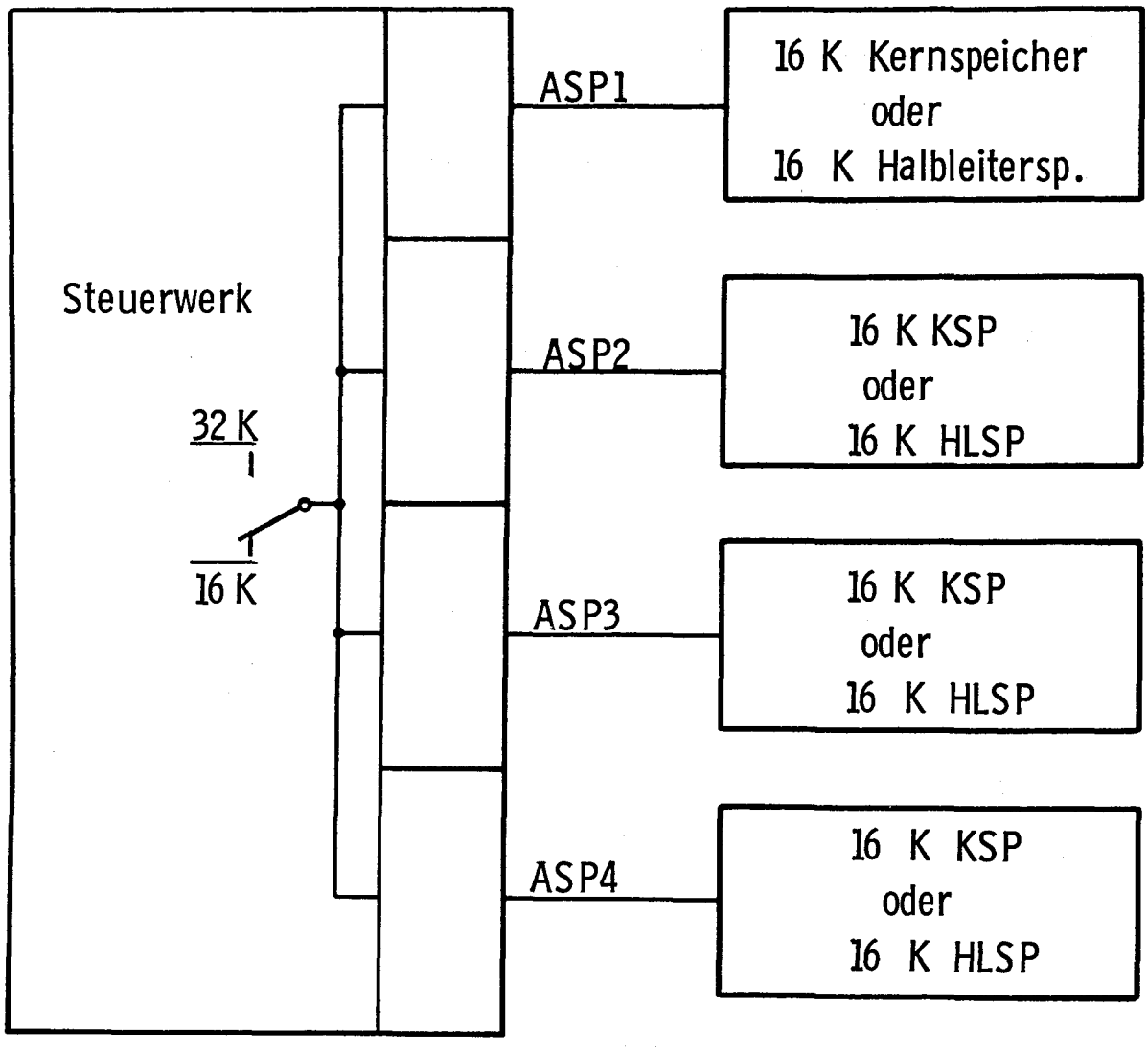
SIEMENS - 306 - Rechnerverbundsystem mit
interaktiven graphischen Sichtgeräten



Stand: April 1980
Angaben in (): Planung
ZEL - Zentrallabor für Elektronik

ABB.: 1

Ausbau 64 K



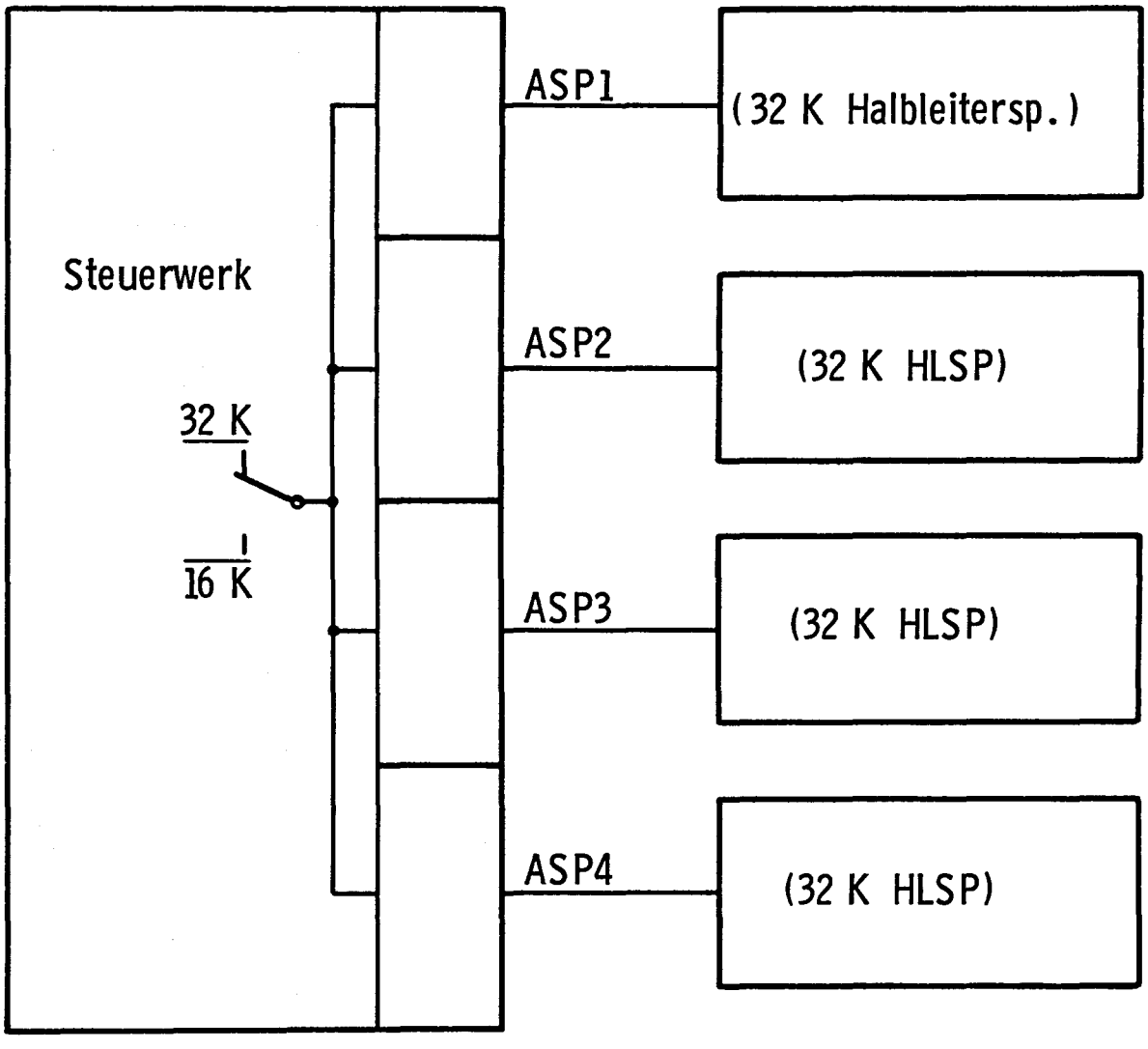
Modifikationen im STW: keine

Material pro Nahtstelle:

1 x Umsetzer 1
1 x Umsetzer 2
1 x HLSP

ABB.: 2

Ausbau 128 K



Modifikationen im STW:

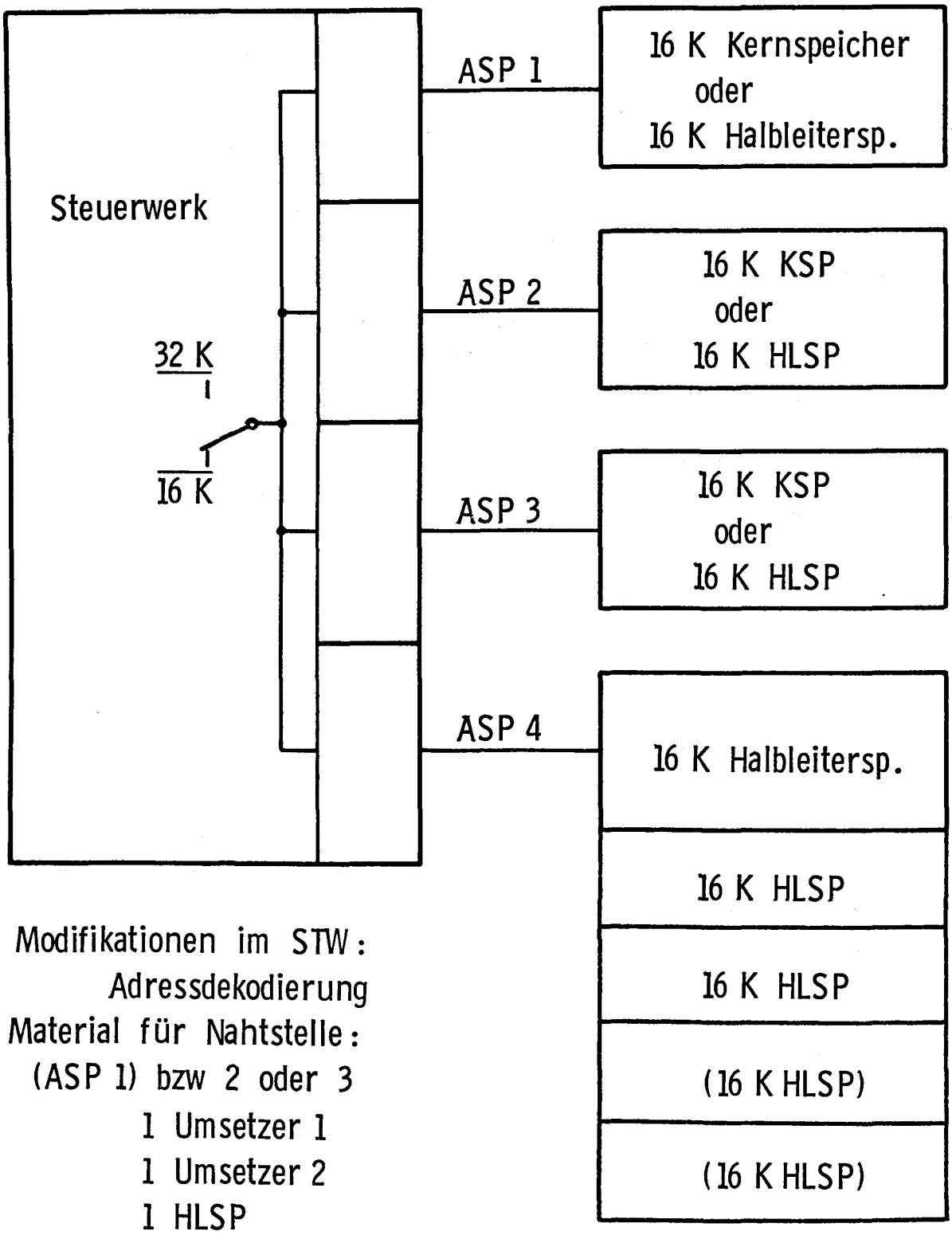
1 Draht für Umschaltung
16 K --- 32 K

Material pro Nahtstelle:

1 Umsetzer 1
1 Umsetzer 2
2 HLSP

ABB.: 3

Ausbau bis 96 K (128 K)



Modifikationen im STW:
Adressdekodierung
Material für Nahtstelle:
(ASP 1) bzw 2 oder 3
1 Umsetzer 1
1 Umsetzer 2
1 HLSP
Material für ASP 4:
1 Umsetzer 1
1 Umsetzer 2
1 - 3 (-5) HLSP

ABB.: 4

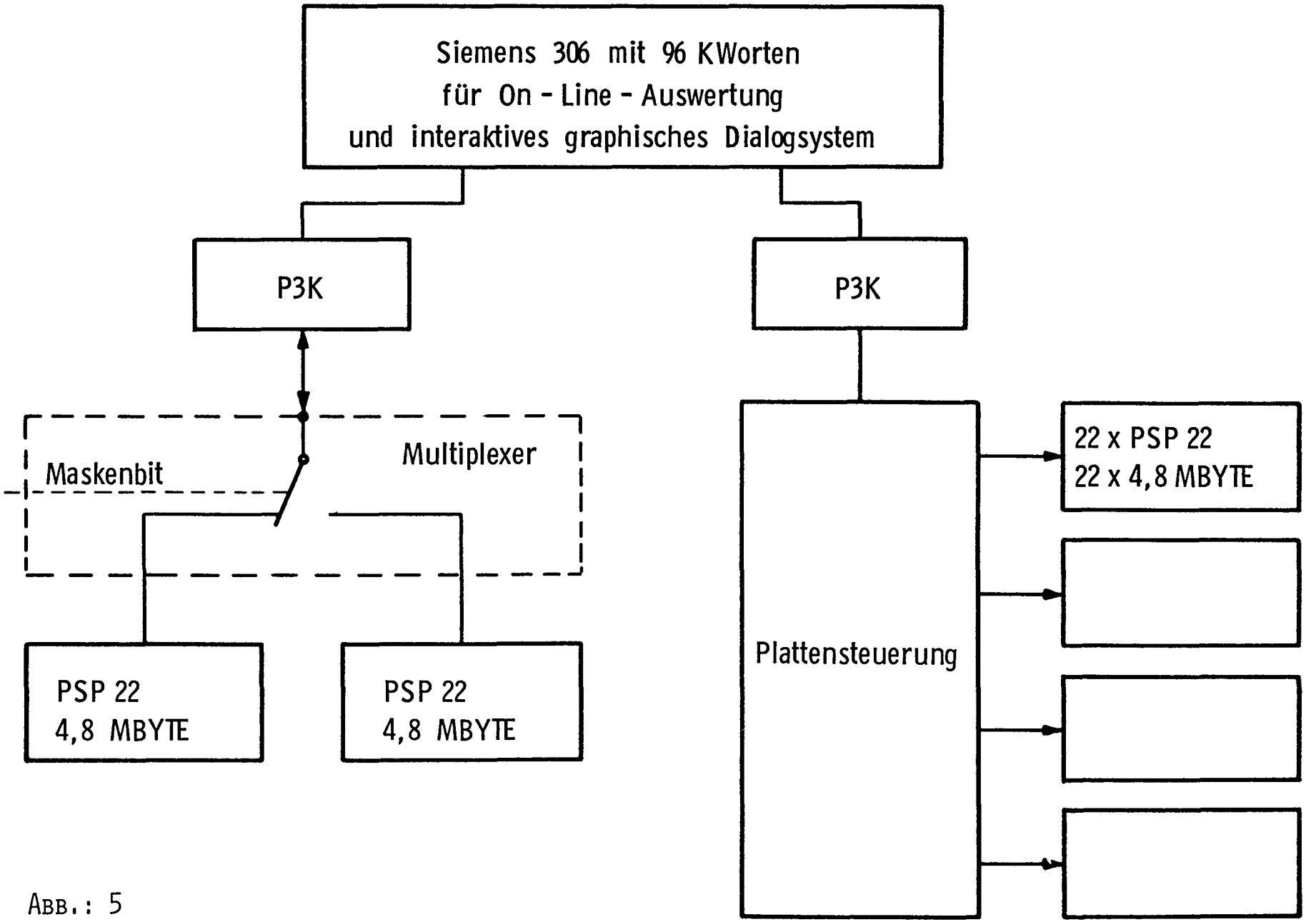

Siemens 306 mit 96 KWorten
für On - Line - Auswertung
und interaktives graphisches Dialogsystem
P3K
P3K
Maskenbit
Multiplexer
PSP 22
4,8 MBYTE
PSP 22
4,8 MBYTE
Plattensteuerung
22 x PSP 22
22 x 4,8 MBYTE

ABB.: 5

F A Q U E L :

Ein interaktives Datenbankanfragesystem

Franz J. Polster
Kernforschungszentrum Karlsruhe
Institut für Datenverarbeitung in der Technik
D-7500 Karlsruhe, Postfach 3640
Bundesrepublik Deutschland

Einleitung

Anläßlich der SAK Jahrestagung 1978 wurde in einem Vortrag /1/ und einer Vorführung das Kleinrechner-Datenbanksystem FADABS vorgestellt. In der Zwischenzeit wurde FADABS sowohl was den Funktionsumfang betrifft als auch zur Erhöhung von Effizienz und Zuverlässigkeit ausgebaut; die wichtigsten Erweiterungen seien kurz aufgeführt:
-- Zur Beschleunigung des Datenretrieval stehen in FADABS jetzt invertierte Dateien zur Verfügung
-- Relationen können nach einem beliebigen Attribut in absteigender oder aufsteigender Ordnung sortiert gelesen werden
-- Die Datenmanipulationssprache FORDAM wurde erweitert
-- FAQUEL, ein interaktives Anfragesystem ist als ein weiteres "FADABS Dienstprogramm" (vergl. /1/) in einer ersten Ausbaustufe fertiggestellt.

Über das interaktive Anfragesystem wird in diesem Beitrag berichtet.

1. FAQUEL: Ziel, Entwurfskriterien

FADABS unterstützte bisher zwei Schnittstellen: die Datendefinitionssprache des Datenbankadministrators und die Datenmanipulationssprache FORDAM des FORTRAN Anwendungsprogrammierers /1/. Zum Lesen, Einfügen und Löschen von Daten mußten also vom Anwender bereitzustellende FORTRAN Programme benutzt werden.

Um einerseits den bekanntermaßen arbeitsintensiven Vorgang der Programmentwicklung vermeiden zu helfen und zugleich auch dem sogenannten "EDV-Laien" /3/ unmittelbar, d.h. ohne vorherige Programmierung, FADABS Datenbanken zugänglich zu machen, wurde das Anfragesystem FAQUEL mit der gleichnamigen Anfragesprache (FADABS QUEry Language) entwickelt /2/. Die Hauptziele beim Entwurf von FAQUEL waren daher:
-- FAQUEL soll einfach erlernbar, verwendbar und leicht verständlich sein
-- FAQUEL soll sowohl Datenretrieval wie auch das Einfügen und Löschen von Daten erlauben
-- FAQUEL soll interaktiv arbeiten
-- Von den Dialoggeräten sollen keine besonderen hardware-Eigenschaften gefordert werden: neben Sichtgeräten sollen auch Blattschreiber eingesetzt werden können.

2. Die Beispiel Datenbank

Relationen, Tupel, Attribute (vergl. /1/):
Eine FADABS Datenbank besteht aus Relationen: "Relation" kann hier mit dem Begriff "Datei" gleichgesetzt werden. Die Datensätze solcher "Dateien" heißen Tupel, eine Relation besteht also aus Tupeln; die (Daten-)Felder von Tupeln heißen Attribute.

In den Abschnitten 3,4 und 5 werden die Möglichkeiten von FAQUEL informal, anhand von Beispielen vorgestellt und erläutert. Es wird hierzu eine fiktive Anwendung aus dem kerntechnischen Bereich angenommen, bei der Kernmaterial in Form von "Plättchen" in einem Lager zu verwalten ist. Die Datenbank besteht aus zwei Relationen:

Die Relation KEMAT mit vier Attributen zur Beschreibung der möglichen Plättchenarten:
-- MATNAM, ein A*10 Attribut, gibt den Materialnamen, hier also den Namen eines Plättchentyps an (A*10: Zeichenkette der Länge 10)
-- Die R*4 Attribute UGES, U235, PUGES bezeichnen (etwa in Gramm) die gesamte Masse Uran bzw. die Masse des U-Isotops 235 bzw. die gesamte Masse Plutonium des Plättchens (R*4: REAL Wert)
-- VERTRG, ein A*2 Attribut, für den Vertrag, unter dem die Plättchen eingeführt worden sind.

Die zweite Relation LAGER mit drei Attributen beschreibt, welche und wieviele Plättchen sich in einer Lagerposition befinden:
-- ORT, ein A*10 Attribut, gibt die Bezeichnung einer Lagerposition an
-- MATNAM, ein A*10 Attribut, gibt wie bei KEMAT den Plättchennamen an
-- ANZAHL, ein I*2 Attribut, liefert die Zahl von Plättchen an der betreffenden Position.

Die Beispiele der nächsten Abschnitte benutzen KEMAT von Bild 2.1 und LAGER von Bild 2.2: so entnimmt man z.B. Bild 2.1, daß ein Plättchen PU NP6 87.85g (0.62g, 31.81g) Uran (U-Isotop 235, Plutonium) enthält und nach Bild 2.2 23 bzw. 18 Plättchen dieser Art in LA/1/1/1 und LA/2/1/9 lagern.

MATNAM	UGES	U235	PUGES	VERTRG
PU NP6	87.85	0.62	31.81	V1
PU VAK NP3	0.0	0.0	55.74	V2
U-35% KP3	124.70	43.64	0.0	V3
U-NAT NB25	1232.00	8.76	0.0	V4
UO2 NP6	127.00	0.90	0.0	V5
PU NPX	103.49	36.12	64.79	V6

Bild 2.1: Die Relation KEMAT

ORT	MATNAM	ANZAHL
LA/1/1/1	PU NP6	23
LA/1/1/1	U-35% KP3	5
LA/1/1/1	PU NPX	31
LA/2/1/9	PU NP6	18

Bild 2.2: Die Relation LAGER

3. Datenretrieval mit FAQUEL

Die FAQUEL Anweisungen zum retrieval (Lesen, Auswerten) von Daten sind allgemein von der Form

LIST <Ausgabe> <Qualifikation> ;

Wie im einzelnen noch genauer erläutert wird, spezifiziert der Benutzer mit "Ausgabe" und "Qualifikation" eine Ergebnisrelation, die dann am Dialoggerät ausgegeben wird.

Die Sprache FAQUEL ist so entworfen, daß sich die Kompliziertheit einer Anfrage in der Komplexität der entsprechenden FAQUEL Anweisung(en) widerspiegelt: Einfache Fragen lassen sich durch einfache LIST-Anweisungen ausdrücken, z.B. ist "Qualifikation" optional und kann falls nicht benötigt entfallen; schwierige Fragen sind i.a. durch längere und aufwendigere LIST- Anweisungen unter Benutzung komplexerer Sprachmittel zu formulieren. Die nachstehenden Beispiele werden dies verdeutlichen, mit einfachsten LIST-Anweisungen beginnend werden zunehmend komplexere vorgestellt.

Beispiel 3.1: Vollständiges Auflisten einer Relation

Anfrage: Die Materialbeschreibungen aller Plättchen werden benötigt!

```
LIST KEMAT;
```

Ergebnis:
Diese Anweisung bewirkt das vollständige Auflisten auf dem Dialoggerät aller Tupel der Relation KEMAT, wie in Bild 2.1 angegeben.

Wird die Ausgabe auf Papier gewünscht, so erreicht man dies mit

```
LIST ON PRINTER KEMAT;
```

In den folgenden Beispielen könnte stets anstelle von "LIST" auch "LIST ON PRINTER" stehen.

Beispiel 3.2: Auflisten von Spalten einer Relation

Anfrage: Welche Plättchen gibt es und wieviel U235 enthält jedes einzelne?

```
LIST MATNAM,U235 OF KEMAT;
```

Ergebnis:

MATNAM	U235
PU NP6	0.62
PU VAK NP3	0.0
U-35% KP3	43.64
U-NAT NB25	8.76
UO2 NP6	0.90
PU NPX	36.12

In den Beispielen 3.1, 3.2 wird nur "Ausgabe" spezifiziert, da jeweils alle Tupel der Relation gewünscht waren. Für die folgenden Anfragen werden Teilmengen, d.h. bestimmte Tupel einer Relation benötigt; diese werden durch eine entsprechende Qualifikation spezifiziert.

Beispiel 3.3:

Anfrage: Welche Plättchen enthalten kein Plutonium?

```
LIST KEMAT WHERE PUGES EQ 0.0;
```

Ergebnis:

MATNAM	UGES	U235	PUGES	VERTRG
U-35% KP3	124.70	43.64	0.0	V3
U-NAT NB25	1232.00	8.76	0.0	V4
UO2 NP6	127.00	0.90	0.0	V5

Ist man wie in Beispiel 3.2 ausschließlich an den Plättchennamen und dem U235-Gehalt dieser Plättchen interessiert, so kann man analog schreiben:

```
LIST MATNAM,U235 OF KEMAT WHERE PUGES LE 0.0;
```

Die Qualifikationen von Beispiel 3.3 bestehen aus je einem Prädikat, wobei jedes einen Vergleich zwischen einem Attributwert eines Tupels und einer Konstanten beschreibt (z.B. "PUGES LE 0.0"). Mit einem Prädikat kann auch ein Vergleich zwischen zwei Attributwerten eines Tupels spezifiziert werden.

Beispiel 3.4:

Anfrage: Welche Plättchen enthalten weniger Uran als Pu und wieviel U235 enthält jedes einzelne?

```
LIST MATNAM,U235 OF KEMAT WHERE UGES LT PUGES;
```

Ergebnis:

```
   MATNAM   |  U235
------------+-------
PU VAK NP3  |   0.0
```

In den bisher angeführten Beispielen bestehen die Qualifikationen aus höchstens einem Prädikat; allgemein können sie analog zu den logical expressions von FORTRAN mehrere Prädikate umfassen, die durch die Booleschen Operatoren OR, AND mit der üblichen Bedeutung miteinander verknüpft sind; ebenso ist (mehrfache) Klammerung erlaubt.

Beispiel 3.5:

Anfrage: Gesucht sind die Namen der Plättchen, die mehr als 500g Uran oder mindestens 100g Uran und mindestens 50g Pu enthalten!

```
LIST MATNAM OF KEMAT
     WHERE UGES GT 500.0    OR
          (UGES GE 100.0   AND  PUGES GE 50);
```

Ergebnis:

```
  MATNAM
----------
U-NAT NB25
PU NPX
```

Noch komplexere Formen von Qualifikationen ergeben sich dadurch, daß in Prädikaten auch Vergleiche mit Mengen von Werten spezifiziert werden können.

Beispiel 3.6:

Anfrage: Gesucht sind die Bezeichnungen der Plättchen mit mehr als 8g U235, die sich in der Position LA/1/1/1 befinden!

```
LIST MATNAM OF KEMAT
     WHERE U235 GT 8.0    AND    MATNAM EQ
           MATNAM OF LAGER WHERE ORT EQ 'LA/1/1/1';        **
```

Ergebnis:

```
  MATNAM
---------
U-35% KP3
PU NPX
```

Erläuterung: Die Qualifikation dieser LIST-Anweisung enthält eine zweite, "geschachtelte" Anfrage (mit ** markiert). Sie liefert als Zwischenergebnis (das nicht ausgegeben wird!) die Namen der Plättchen, die in LA/1/1/1 lagern, hier also: PU NP6, U-35% KP3, PU NPX; der Vergleichsoperator EQ, der sich damit auf eine Menge von Werten bezieht, bedeutet in dieser Situation: "Es ist ein Element in der Menge enthalten, das gleich (EQ) dem Attribut MATNAM ist". Es werden also die Materialbezeichnungen aller KEMAT-Tupel gelistet, bei denen U235 größer als 8 und MATNAM gleich einem der drei genannten Plättchennamen ist.

Beispiel 3.7: Verkettung zweier Relationen (JOIN-Operation)

Anfrage: Zu jedem Ort sind die Verträge der Materialien anzugeben, die dort lagern!

```
LIST ORT OF LAGER   JOINED TO   VERTRG OF KEMAT
                BY MATNAM OF LAGER  EQ  MATNAM OF KEMAT;
```

Ergebnis:

```
  ORT     | VERTRG
----------+-------
LA/1/1/1  |   V1
LA/2/1/9  |   V1
LA/1/1/1  |   V3
LA/1/1/1  |   V6
```

Erläuterung: Es wird jedes Tupel von LAGER mit den Tupeln von KEMAT verkettet, die in MATNAM (von KEMAT!) den gleichen Wert haben. Von der so erzeugten (Zwischen-)Relation werden ORT und VERTRG aufgelistet. Eine ausführlichere Darstellung enthält /4/.

4. Ändern der Datenbank mit FAQUEL

Neben dem Datenretrieval ermöglicht FAQUEL das Einfügen bzw. Löschen von Tupeln.

Beispiel 4.1: Einfügen von Daten über das Dialoggerät

An die Position LA/2/5/2 sind 15 Plättchen PU VAK NP3 transportiert worden; es ist also ein entsprechendes Tupel in LAGER einzufügen:

```
INSERT INTO LAGER: < 'LA/2/5/2', 'PU VAK NP3', 15 >;
```

Ergebnis: Nach dem Einfügen der in < und > eingeschlossenen Attributwerte enthält LAGER folgende Tupel:

ORT	MATNAM	ANZAHL
LA/1/1/1	PU NP6	23
LA/1/1/1	U-35% KP3	5
LA/1/1/1	PU NPX	31
LA/2/1/9	PU NP6	18
LA/2/5/2	PU VAK NP3	15

Beispiel 4.2: Einfügen von Daten der Datenbank in eine Relation

LAGNAM sei eine weitere Relation mit nur einem Attribut für Namen von Lagerpositionen; LAGNAM enthalte noch keine Tupel. Mit

```
INSERT INTO LAGNAM: ORT OF LAGER WHERE MATNAM EQ 'PU NP6';
```

werden die Positionen in LAGNAM eingetragen, an denen Plättchen vom Typ PU NP6 lagern: LAGNAM enthält dann also: LA/1/1/1, LA/2/1/9.

Beispiel 4.3: Löschen von Daten der Datenbank

Es seien alle Plättchen, die kein Pu enthalten, ausgelagert worden, so daß deren Beschreibungen nicht mehr benötigt werden. Mit folgender Anweisung kann die Relation KEMAT auf den aktuellen Stand gebracht werden:

```
DELETE IN KEMAT  WHERE PUGES EQ 0.0;
```

Ergebnis: KEMAT enthält dann noch folgende Tupel:

MATNAM	UGES	U235	PUGES	VERTRG
PU NP6	87.85	0.62	31.81	V1
PU VAK NP3	0.0	0.0	55.74	V2
PU NPX	103.49	36.12	64.79	V6

5. Der Tupelmodus

FAQUEL läßt sich mit "TUPLEMODE ON;" aus dem Standardmodus S-Modus in den sogen. Tupel-Modus versetzen. Mit "TUPLEMODE OFF;" kehrt man wieder in den S-Modus zurück.

Im S-Modus kann der Benutzer erst nach Abschluß einer LIST-Anweisung eine neue Anweisung eingeben. In der Praxis ist es jedoch häufig erforderlich, daß der Benutzer nach der Ausgabe eines jeden Tupels die Möglichkeit hat "nachzuhaken", d.h. eine neue Anfrage zu formulieren, zeitlich parallel zur laufenden aktuellen Anfrage; genau dies erlaubt FAQUEL im Tupelmodus:

FAQUEL befinde sich im Tupelmodus und die aktuelle Anfrage sei

```
LIST MATNAM,U235 OF KEMAT WHERE PUGES LE 0.0;
```

Das Tupel

```
U-NAT NB25    8.76
```

sei gerade ausgegeben worden; es ist damit das aktuelle Tupel. Der Benutzer hat jetzt die Möglichkeit,

-- das nächste Tupel anzufordern

-- weitere Attributwerte des aktuellen Tupels, z.B. VERTRG, anzufordern: Mit der Anweisung "GIVE VERTRG;" wird also in diesem Fall "V4" ausgegeben (vergl. Bild 2.1).

   Diese Möglichkeit ist für die Praxis von besonderer Bedeutung, da dort Relationen von bis zu 40 Attributen keine Seltenheit sind und man solche Tupel vollständig und übersichtlich in einer Ausgabezeile (80 bzw. 132 Zeichen/Zeile) gar nicht auflisten kann!

-- das aktuelle Tupel zu löschen mit "DELETE;"

-- Attributwerte des aktuellen Tupels zu ändern

-- ein neues Tupel einzufügen, bei dem einige Attributwerte mit Attributen des aktuellen übereinstimmen und andere explizit eingegeben werden

-- eine weitere Anfrage zu formulieren; diese wird dann zur aktuellen Anfrage und ebenfalls im Tupelmodus abgearbeitet. Dies kann im Prinzip beliebig häufig wiederholt werden.

6. Zusammenfassung, Stand der Arbeiten

Mit FAQUEL steht ein System zu Verfügung, das es dem EDV-Laien auf einfache Weise erlaubt, auf FADABS Datenbanken zuzugreifen.

Es konnten hier nur die wichtigsten Anweisungen und Möglichkeiten von FAQUEL anhand von Beispielen vorgestellt werden, eine vollständige und formale Beschreibung zusammen mit dem Systementwurf findet man in /2/: Dort sind weitere Anweisungen vorgesehen, z.B. zum Ablegen von Zwischenergebnissen in "Zwischenrelationen", zum Sortieren der Ausgabe nach einem beliebigen Kriterium.

Die zur Zeit (April 1980) verfügbare FAQUEL Version 1.0 realisiert LIST-Anweisungen, wie sie in den Beispielen 3.1, 3.2, 3.3, 3.5 gezeigt worden sind. An der Implementierung der update-Anweisungen (Abschnitt 4) und des Tupelmodus (Abschnitt 5) wird gegenwärtig gearbeitet.

FAQUEL ist als weiteres FADABS Dienstprogramm (vergl. /1/) ein selbständiges Programm im Sinne des Betriebssystems: es benötigt segmentiert ca. 20 KW Hauptspeicher und setzt den FADABS Nucleus voraus. FAQUEL ist wie das gesamte Datenbanksystem FADABS auf SIEMENS Prozeßrechnern 330 und R30 installiert und verfügbar.

Referenzen:

/1/ Polster, F.J.: FADABS: Ein Datenbanksystem für die SIEMENS Prozeßrechner S330. Tagungsbericht, 9. Jahrestagung des SIEMENS-Anwenderkreises I, Kernforschungszentrum Karlsruhe, 1978, KfK-Bericht 2642

/2/ Tritschler, P.: FAQUEL: Eine interaktive Anfragesprache für ein relationales Datenbanksystem. Diplomarbeit, Universität Karlsruhe, Fakultät für Informatik, Juni 1979

/3/ Wedekind,H.: Datenbanksysteme I. Reihe Informatik/16, Bibliographisches Institut, Mannheim 1974

/4/ Polster, F.J.: FAQUEL: Ein interaktives Datenbankanfragesystem. Kernforschungszentrum Karlsruhe, Primärbericht 09.01.02P11C, April 1980.

RECHNERKOPPLUNG ZWISCHEN SIEMENS PR 330 UND ICL 1906 S

J.A.V. Thornton, MA
Dipl. Math. W. Schmid
Institut für elektrische Maschinen, Antriebe und Bahnen
TU Braunschweig

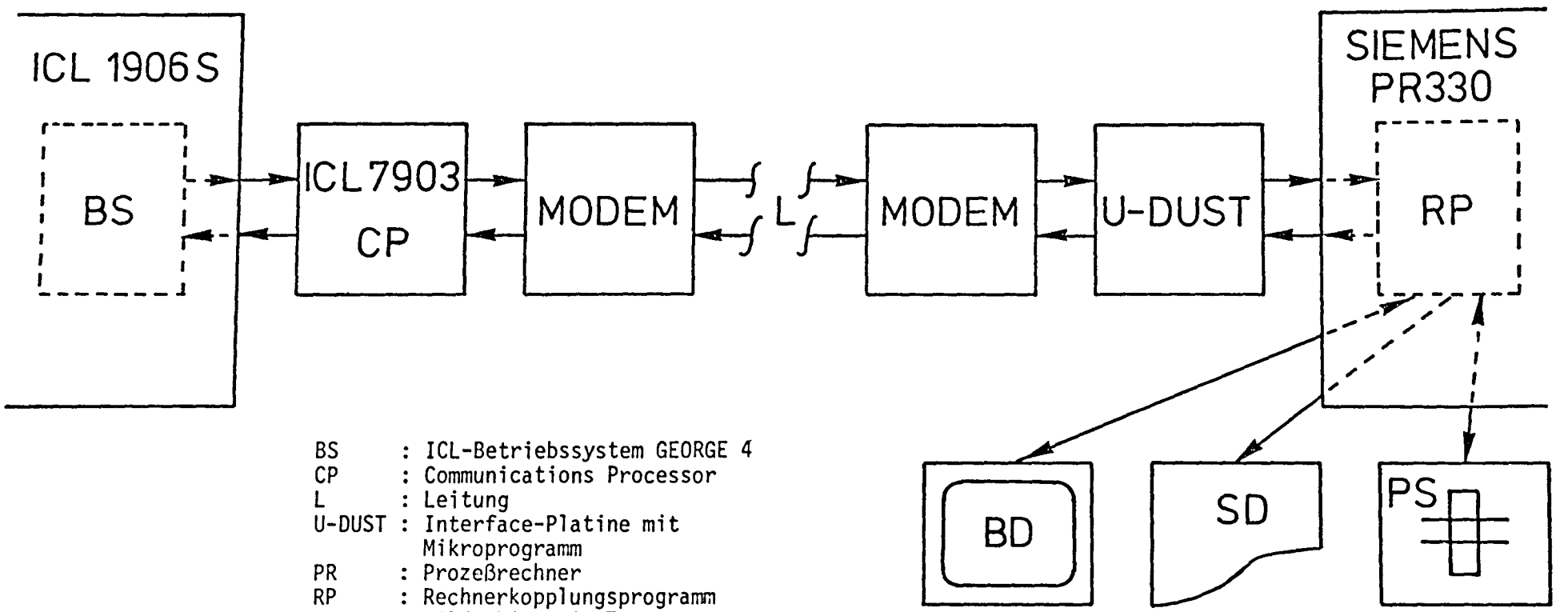
Dr. E.E. Pohl*
Siemens AG Braunschweig

1. Obersicht

Dieser Beitrag beschreibt eine Rechnerkopplung zur blockweisen, synchronen Obertragung von Daten zwischen einem ICL-1906 S-Rechner und einem Siemens PR 330. Ein plattenspeicherresidentes Programm im PR 330 simuliert die Funktionen einer "Remote Job Entry Station", d.h. einer Datenstation mit mehreren peripheren Geräten. On-line Dialogbetrieb ist nicht möglich. Bild 1 zeigt die Konfiguration in seiner einfachsten Form, wobei nur eine Leitung an dem ICL-Daten-Übertragungsrechner (CP) angeschlossen ist. Die Daten werden durch Modem über eine 4800 Bit/s Leitung zum und vom PR 330 übertragen. Die Datenübertragung erfolgt im Halb-Duplex-Modus, d.h. zu einem gegebenen Zeitpunkt kann nur in einer Richtung übertragen werden. Im allgemeinen können mehrere Stationen an eine Leitung angeschlossen werden, so daß eine Datenaustauschprozedur (Protokoll) notwendig wird, die eine vorherige Adressierung der jeweiligen Datenstation vor Einleitung des Datenaustausches ermöglicht. Diese Prozedur entspricht einer Untermenge der "ISO-Control-Procedures". Der Unterschied zur MSV 2 - Prozedur besteht hauptsächlich darin, daß sowohl die Adressierung einer Station auf der Leitung als auch der nachfolgende Anschluß eines peripheren Gerätes auf andere Weise erfolgt. Diese Unterschiede machten die Entwicklung einer besonderen Schnittstelle zum PR 330 notwendig. Dafür wurde die universelle Datenübertragungssteuerung (U-DUST), eine Datenübertragungssteuerung mit mikroprogrammierter Ablaufsteuerung, eingesetzt. Ihr Aufbau und ihre Arbeitsweise werden in einem späteren Abschnitt kurz erläutert.

---

*Dr. Pohl war während der Entstehung der Kopplung wiss. Assistent am Institut für elektrische Maschinen, Antriebe und Bahnen der TU Braunschweig



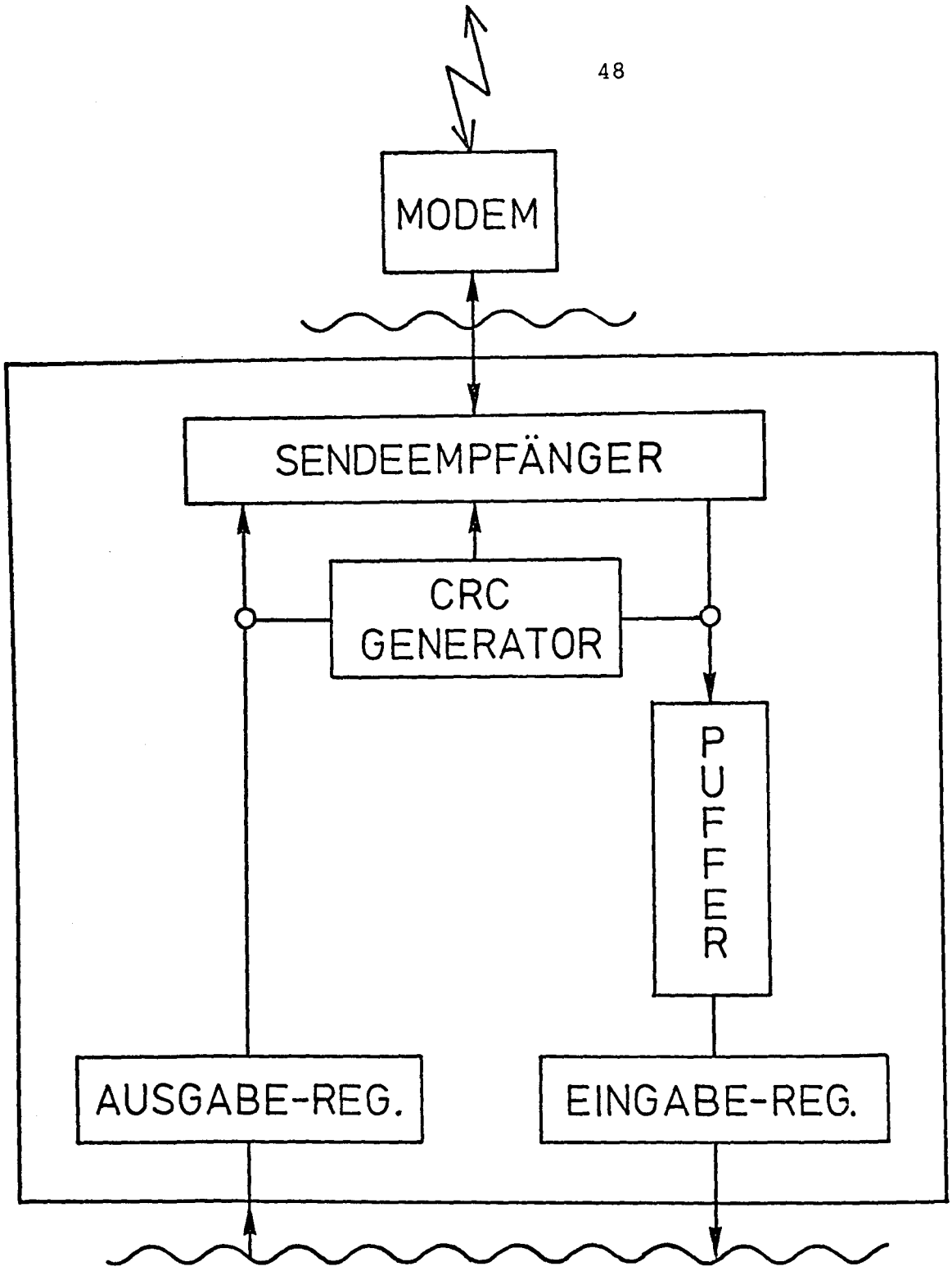
BS : ICL-Betriebssystem GEORGE 4
CP : Communications Processor
L : Leitung
U-DUST : Interface-Platine mit Mikroprogramm
PR : Prozeßrechner
RP : Rechnerkopplungsprogramm
BD : Bildschirm mit Tastatur
SD : Schnelldrucker
PS : Plattenspeicher

Bild 1: Konfiguration des Rechnerkopplungssystems

Aus organisatorischen Gründen war es nicht sinnvoll, den vollen Ablauf der Obertragungsprozedur durch das U-DUST-Mikroprogramm zu erfassen, da bei jeder Quittung aktuell zu bildende Statusbits mit übergeben werden müssen. Der übergeordnete Ablauf der Datenübertragung wird siemensseitig durch das plattenspeicherresidente Rechnerkopplungsprogramm übernommen. Dieses Programm ist auch für die Koordinierung des Datenstroms mit den jeweiligen angeschlossenen peripheren Geräten zuständig, wobei der Anschluß der Geräte durch Benutzer-Tastaturbefehle gesteuert werden kann.

2. Die mikroprogrammierte Obertragungssteuerung

Die Daten werden bitseriell in Blöcken von max. 80 Zeichen auf der Leitung übertragen. Eine Hauptaufgabe der U-DUST besteht also darin, den bitseriellen Datenstrom an der Modemschnittstelle an die bitparallele PR 330-Schnittstelle anzupassen.
Der Befehlsvorrat der Mikroprogrammablaufsteuerung ermöglicht außerdem weitere Funktionen. Zum einen kann die Richtigkeit der Blockformate durch die am Anfang und am Ende des Blocks befindlichen Steuerzeichen geprüft werden. Zum anderen können Paritätszeichen geprüft werden, sowohl die Querparität für jedes einzelne Zeichen als auch die Längsparitätdes ganzen Datenblocks durch Zuhilfenahme des CRC-Characters. Bei der Erkennung von Fehlern können fehlerspezifische Betriebsanzeigen an das Rechnerkopplungsprogramm übergeben werden, das anschließend von der Obertragungsprozedur vorgeschriebene Maßnahmen treffen kann.
Es wurde bereits erwähnt, daß es mehrere Datenstationen auf einer Leitung gibt. Bei jedem empfangenen Datenblock muß entschieden werden, ob er für diese Station bestimmt ist. Wenn dieses ständig durch das PR 330-Rechnerkopplungsprogramm geprüft werden müßte, würde eine zusätzliche Belastung der PR 330-CPU entstehen, ohne daß die Rechnerkopplung dafür effektiv ausgenutzt würde. Dieses Problem wird durch die Verwendung der auf der U-DUST-Platine befindlichen Eingabepuffer gelöst. Ankommende Datenblöcke werden zunächst im Eingabepuffer aufbewahrt, und sie werden erst dann an das Rechnerkopplungsprogramm übergeben, wenn das U-DUST-Mikroprogramm festgestellt hat, daß sie für diese Datenstation bestimmt sind.
Wir können uns jetzt ein Bild des Datenflusses in der U-DUST machen (Bild 2). Gesendete Datenblöcke werden zeichenweise in das U-DUST-Ausgaberegister weitergegeben. Anschließend gelangt das Zeichen



PR 330
(RECHNERKOPPLUNGSPROGRAMM)

Informationsfluß in der U-DUST

Bild 2

in den Sendeempfängerbaustein (hier wird die Querparität gebildet), und es wird als eine Bitfolge an das Modem geschickt. Gegebenenfalls wird ein generiertes CRC-Zeichen am Ende des Blocks ausgesendet. Empfangene Blöcke werden durch den Sendeempfänger zu Zeichen zusammengestellt, wobei die Querparität geprüft wird. Die so entstandenen Zeichen werden in einem Eingabepuffer aufbewahrt, bis der Datenblock vollständig empfangen worden ist. Gegebenenfalls wird die Längsparität durch den CRC-Generator geprüft. Anschließend wird der Datenblock über das Eingaberegister an das Rechnerkopplungsprogramm übergeben. Die Verwendung des Eingabepuffers führt dazu, daß die Oberprüfung des Datenblocks durch die U-DUST vom Rechnerkopplungsprogramm entkoppelt wird, was zur bereits erwähnten Entlastung der CPU beiträgt.

Da ein Assembler für die U-DUST-Mikroprogrammsprache nicht zur Verfügung stand, wurde ein eigener Assembler entwickelt, der umfangreiche Syntaxprüfungen des Mikroprogramms durchführt, so daß die Entwicklungsarbeit erheblich erleichtert wurde. Der Assembler wurde in PASCAL (ICL) geschrieben, eine Sprache, die sich sehr gut für Textverarbeitungsprobleme einsetzen läßt.

3. Das Rechnerkopplungsprogramm

Das Rechnerkopplungsprogramm wurde in der Siemens Assemblersprache ASS 300 geschrieben und beansprucht 5K Worte Speicherplatz. Zur Steuerung der U-DUST wird die PR-330-Rechnerkopplungsschnittstelle verwendet. Die Entkopplung durch die U-DUST ermöglichte es, daß das Rechnerkopplungsprogramm plattenspeicherresident ausgelegt werden konnte. Wie bereits erwähnt, sorgte das Programm für den übergeordneten Ablauf der Übertragungsprozedur, unter anderem für das Verhalten beim Auftreten von Fehlern, so daß z.B. eine Wiederholung eines eventuell fehlerhaft empfangenen Datenblocks durch eine entsprechende Rückmeldung (+ Statusbits) über die Rechnerkopplung angefordert werden kann.

Die Übertragungsprozedur sieht den Anschluß von verschiedenen Geräten vor: Bildschirm, Tastatur, Schnelldrucker, Kartenleser, Streifenstanzer und Streifenleser, wobei nur jeweils eine Zeile von der Tastatur bzw. zum Bildschirm übertragen werden kann. Der Anschluß eines Kartenlesers wurde nicht implementiert. Die beiden Lochstreifengeräte werden durch Dateien simuliert, deren Namen durch Benutzersteuerbefehle angegeben werden können. Dadurch konnte eine schnelle und einfache Übertragung von Dateien zwischen der ICL und dem PR 330 gewährleistet werden.

Das Programm nimmt einzelne Tastaturzeilen als Eingabe an. Zeilen, die nicht mit dem Sonderzeichen "*" beginnen, werden an die ICL übertragen. Dort werden sie vom Betriebssystem als ICL-Befehle interpretiert. Zeilen, die am Anfang "*" enthalten, sind Steuerbefehle für das Programm. Diese Befehle steuern den Anschluß der Geräte an der Rechnerkopplung und lassen zusätzlich einen Abbruch bzw. ein Anhalten einer laufenden Obertragung zu. Um eine Datei z.B. an die ICL zu übertragen, wird sie durch Angabe ihres Namens als "virtuelles Gerät" mit der Rechnerkopplung verbunden. Nach anschließender blockweiser Obertragung des Dateiinhaltes wird die Datei geschlossen, und das "Gerät" wird von der Rechnerkopplung abgekoppelt. Bildschirmmeldungen informieren den Benutzer über den Verlauf der Obertragung. Eventuell werden Fehlermeldungen ausgegeben.

Zusätzlich zu den normalen Steuerbefehlen wurden auch Testbefehle vorgesehen. Diese Befehle geben Aufschluß über die Struktur der übertragenen Datenblöcke und erwiesen sich während der Rechnerkopplungserprobungsphase als sehr hilfreich.
Das Rechnerkopplungsprogramm wurde so weit wie möglich in einer modularen Unterprogrammstruktur geschrieben. Damit konnte der weiteren Entwicklung des Rechnerkopplungssystems im Hinblick auf neue Datenprotokolle Rechnung getragen werden.

4. Ausblick

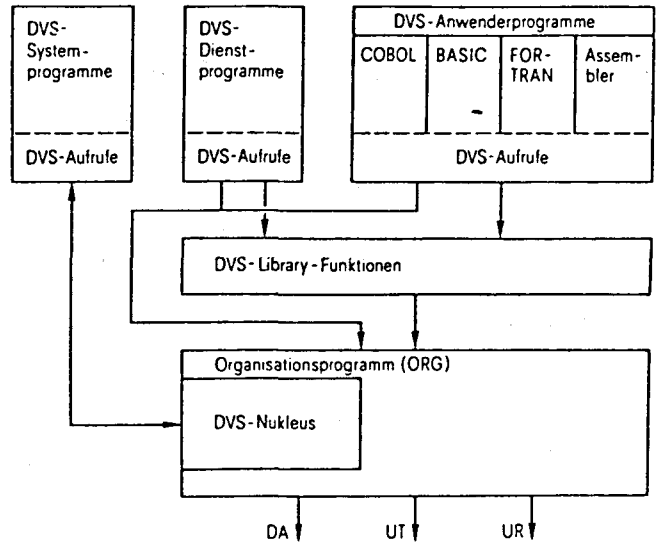
Obwohl die Rechnerkopplung aufgrund der durch die Leitung festgesetzten Einschränkungen für eine Obertragungsrate von 4800 Bit/sec ausgelegt wurde, ist das System bei 9600 Bit/sec auch erfolgreich getestet worden. Das System wird augenblicklich auf ein neues Datenprotokoll umgestellt und befindet sich in der Erprobungsphase. Das neue Protokoll ermöglicht unter anderem den Anschluß von einer fast unbegrenzten Anzahl von peripheren Geräten, wobei Bildschirmdialogbetrieb an der ICL auch zugelassen wird. Hierbei werden Datenblöcke von maximal bis zu 256 Zeichen übertragen. Dieses stellt kein Problem dar, da der Eingabepuffer für diese maximale Länge ausgelegt ist. Durch Änderung des Mikroprogrammes konnte die U-DUST wahlweise für dieses neue Protokoll und für das der vorhergehenden Abschnitte ausgelegt werden und zwar so, daß die Umschaltung softwaremäßig durch das Rechnerkopplungsprogramm erfolgen kann, ohne daß ein Umtausch von Platinen oder Bausteinen notwendig wird.
Wir hoffen, daß es zu einem späteren Zeitpunkt möglich sein wird, über die Ergebnisse dieser neuen Entwicklung zu berichten.

Datenverwaltungssystem DVS 300

Im Bereich der industriellen Automatisierungstechnik bis zur Datentechnik im Produktionsbereich sind heute umfangreiche Datenmengen zu verarbeiten.

Eine einfache Dateiorganisation ist häufig nicht mehr ausreichend. Gefordert wird ein Datenverwaltungssystem, das Bestandteil des Betriebssystems ist, Systemleistungen für Datenschutz und Datensicherheit erbringt und auch Maßnahmen für Daten- und Programmportabilität vorsieht.

Diese Forderungen erfüllt im Rahmen der Siemens-Systeme 300 das modular strukturierte Datenverwaltungssystem DVS 300. Mit seiner zentralen Stellung im Siemens-System 300 und mit seinen leistungsstarken Funktionsmöglichkeiten ist DVS 300 ein standardisiertes Anwendungs-Systemprogramm zur beliebigen Verwaltung von Daten. DVS 300 löst die Datenverwaltungsaufgaben für die Siemens-Systeme R10, R20, R30 und R40 ausbaubar und zukunftsicher.



DA direct access (z. B. Magnetplatte) UR unit record (z. B. Drucker)
UT unit tape (Magnetband)

Bild 1 Struktur des Datenverwaltungssystems DVS 300

Architektur

Struktur des Systems DVS 300

Aufgrund unterschiedlichster Anwendungsfälle werden zahlreiche Anforderungen an ein Datenverwaltungssystem gestellt. Dazu gehören u. a. Modularität und aufgabenspezifische Funktionsanpassung. Aus diesem Grund wurde das System DVS 300 schichtweise aufgebaut (Bild 1). Die drei Hauptkomponenten sind Nukleus, Library-Funktionen sowie System- und Dienstprogramme.

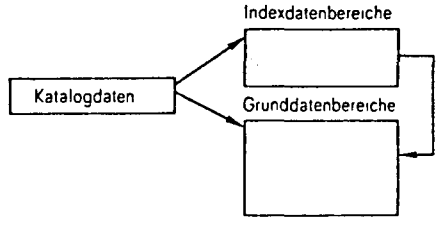


Bild 2
Struktur eines Files bei Direktzugriffsspeichern

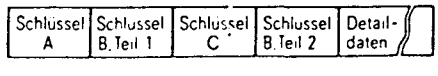


Bild 3
Schlüssel im Grunddatensatz

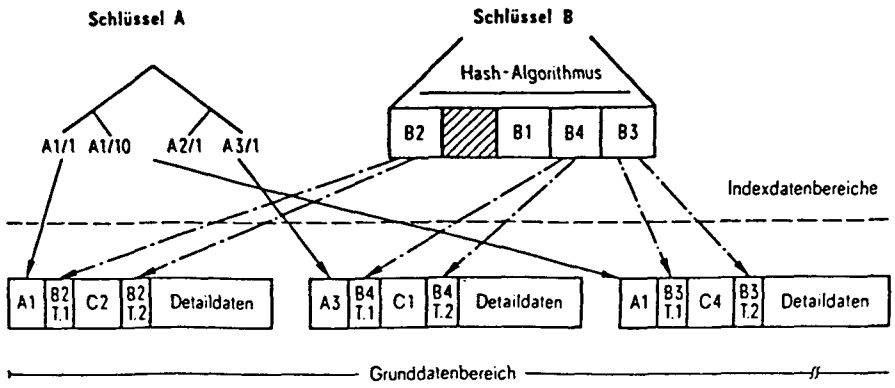


Bild 4
Beispiel eines dreifach indizierten Files. Die Sätze im Grunddatenbereich sind bezüglich Schlüssel C nach Hash-Algorithmus abgelegt

Der Nukleus bearbeitet alle Zugriffsfunktionen und koordiniert den simultanen Zugriff mehrerer Programme auf gleiche Datenbestände oder Betriebsmittel. Die Integration des Nukleus in die Betriebssysteme 300 erzielt ein zeitoptimiertes Zugriffsverhalten.

Als Ergänzung der Nukleus-Funktionen steht eine DVS-300-Library-Funktion zur Verfügung. Eine anwendungsspezifische Funktionsanpassung ist über eine einfach zu handhabende Library-Erweiterung gegeben.

Werden nur bestimmte DVS-300-Funktionen benötigt, so kann der DVS-300-Anwender einen für seine Aufgaben geeigneten Nukleus und eine hierauf abgestimmte Library-Funktion erstellen. Diese DVS-300-Generierung wird durch den modularen Systemaufbau ermöglicht. Sie ist benutzergerecht ausgelegt und reduziert den notwendigen Speicherplatzbedarf auf ein Minimum.

Ein umfangreiches Paket von DVS-300-System- und -Dienstprogrammen rundet die Leistungen von DVS 300 ab. Die wesentlichen Funktionen dieser Programme können sowohl durch Aufrufe aus den Anwenderprogrammen als auch durch Kommandos der leistungsstarken Bedienschnittstelle angesprochen werden.

Im einzelnen sind dies:

- Einrichten und Löschen von Files (Dateien),
- Abfragen und Ändern von File-Eigenschaften,
- Reorganisieren und Kopieren von Files,
- formatgesteuertes Ein- und Ausgeben von Daten mit Konvertierungsroutinen,
- Rekonstruieren von Datenbeständen anhand von Informationen der Sicherungsdatenträger,
- Einbringen und Wechseln von DVS-300-Datenträgern über Betriebskommandos.

DVS-300-Datenstrukturen

Vorteilhaft für den Benutzer des Systems DVS 300 ist die einheitliche Darstellung der Files (Bild 2). Jeder DVS-300-File kann sich bei Verwendung von Direktzugriffsspeichern in zwei Bereiche, Grunddatenbereich und Indexdatenbereich, gliedern. Die Zuordnung dieser Bereiche und die Beschreibung der File-Eigenschaften sind in den Katalogdaten hinterlegt.

Grunddatenbereich

Der Grunddatenbereich enthält die zu verwaltenden Anwenderdaten. Ein Vorzug von DVS 300 liegt in dem einheitlichen Aufbau der Grunddatenbereiche, unabhängig von der jeweils verwendeten Organisationsform. Aufgrund dieses Aufbaus ist der Übergang von fester zu variabler Satzlänge ohne Neuladen der Daten jederzeit möglich. Je nach Anwendungsfall können die Grunddatenbereiche eingangssequentiell, über Freispeicherverwaltung, über Satznummer oder nach Hash-Algorithmus abgelegt werden. Zur optimalen Ausnutzung des zur Verfügung stehenden Speicherplatzes kann jeder Grunddatenbereich über mehrere gleichartige Datenträger verteilt sein (Multi-volume-File).

Indexdatenbereich

Sollen Datensätze des Grunddatenbereichs nach Suchbegriffen schnell ausgewählt werden, dann können im Grunddatensatz Felder als Schlüssel vereinbart werden. Da die Anordnung der Felder nicht immer mit den benötigten Schlüsseln übereinstimmt, lassen sich durch beliebige Kombination der Felder neue Schlüssel festlegen (Bild 3).

Jeder Indexdatenbereich liegt geschlossen auf einem Datenträger und kann wahlweise als B*-Baum (entspricht ISAM) oder nach HASH abgelegt werden. Um die Zuordnung von einem Schlüsselwert zu mehreren Datensätzen zu ermöglichen – z.B. von der Nummer eines Kunden zu dessen Aufträgen –, wurde die B*-Baum-Verwaltung erweitert. Entscheidend ist jedoch für den DVS-300-Benutzer, daß die Indexdatenbereiche unabhängig von der gewählten Verwaltung keine zusätzlichen Reorganisationsläufe erfordern.

Bild 4 zeigt einen dreifach indizierten File. Der ISAM-Schlüssel A (mit Duplikaten A1/1 und A1/10) ist als B*-Baum angelegt. Der Indexdatenbereich des eindeutigen Schlüssels B wird nach einer HASH-Funktion verwaltet und besteht zudem aus zwei Schlüsselteilen. Die Grunddaten selbst sind bezüglich Schlüssel C nach Hash-Algorithmus angeordnet (HASH-B).

DVS-300-Zugriffe

Die zur Speicherung von Daten ausgewählte Datenstruktur bietet eine oder mehrere Zugriffsmöglichkeiten auf diese Daten. Durch die jeweils geeignetste Zugriffsmöglichkeit lassen sich erhebliche Steigerungen der Verarbeitungsgeschwindigkeit erreichen.

In DVS 300 stehen im einzelnen als Zugriffsmöglichkeiten zur Verfügung:

- SEQ physikalisch/logisch, sequentiell vorwärts und rückwärts,
- REL über Satznummer,

ISAM, HASH und HASH-B über Schlüssel.

Ergänzt werden diese Zugriffsmöglichkeiten durch Aufrufe zum Positionieren innerhalb eines Files.

Organisatorische Funktionen des DVS 300

Neben den Zugriffs- und Positionierungsfunktionen stellt DVS 300 eine Reihe von organisatorischen Funktionen zur Verfügung.

Bevor auf einen File zugegriffen werden kann, muß dieser dem System DVS 300 bekanntgemacht werden. Diese Aufgabe übernimmt die CREATE-Funktion unter Angabe aller File-Eigenschaften, wie Datenträger, Benutzerkennzeichen, Paßwort oder Schlüsseldefinitionen. Bei Veränderungen von File-Eigenschaften, z.B. Hinzunahme weiterer Schlüssel oder Verlängerung des Files, kann ebenfalls diese Funktion verwendet werden. Wird ein File nicht mehr benötigt, so kann es mit der ERASE-Funktion gelöscht werden.

Die Verbindung zwischen DVS-300-Anwenderprogramm und den zu bearbeitenden Files stellt die OPEN-Funktion her. Hierbei werden zahlreiche Prüfungen, z.B. auf zulässiges Benutzerkennzeichen, Paßwort, Eröffnungsmodus oder zulässige Zugriffe, durchgeführt. Den Abschluß der File-Bearbeitung kennzeichnet die CLOSE-Funktion.

Ist bei der Programmerstellung noch ungeklärt, auf welchem Datenträger der File abgelegt oder wie er betrieben werden soll, dann kann bei der OPEN-Funktion ein Hinweis auf fehlende Größen hinterlegt werden. Dadurch werden erst zum Bearbeitungszeitpunkt der OPEN-Funktion mit Hilfe der LINK-Funktion die fehlenden Größen automatisch ersetzt. Aktuelle Gegebenheiten lassen sich damit ohne Änderung des Anwenderprogramms berücksichtigen.

Das Spektrum der organisatorischen Funktionen umfaßt schließlich die REORG-, COPY- und INQUIRE-Funktion. Mit der REORG-Funktion läßt sich u.a. eine Komprimierung des Grunddatenbereichs, nachträgliche Aktualisierung von Indexdatenbereichen oder die Wiederherstellung zerstörter Verwaltungsdaten erreichen. Werden Duplikate von Grunddatenbereichen und Indexdatenbereichen benötigt, so übernimmt dies die COPY-Funktion. Die INQUIRE-Funktion gibt Auskunft über File-Eigenschaften.

Allen organisatorischen Funktionen ist gemeinsam, daß sie im Anwenderprogramm aufrufbar sind. Die Funktionen CREATE, ERASE, REORG, COPY und INQUIRE verfügen zusätzlich über eine Bedienungsschnittstelle.

Transferoptimierung

Im System DVS 300 werden beim Datenzugriff Puffer benutzt. Die jeweils notwendige Transferanzahl wird bei Files auf Direktzugriffsspeichern durch ein Daten-Paging begrenzt. Bereits im Puffer stehende Daten werden erkannt; entsprechende Transfers unterbleiben. Die Leistungsfähigkeit dieses Daten-Paging hängt dabei in entscheidendem Maße von der zur Verfügung gestellten Anzahl der Puffer ab.

Die Aufzeichnungsleistung auf Magnetband kann vom Benutzer durch geeignete Wahl des Blockungsfaktors beeinflußt werden. Ein wahlweiser Wechselpufferbetrieb steigert zudem die Performance (Datendurchsatz je Zeiteinheit).

Neben der Transferoptimierung durch systeminterne Verfahren kann auch der Benutzer das Zugriffszeitverhalten von DVS 300 mitbeeinflussen, z.B. durch Ausschluß bestimmter Schlüssel von der laufenden Aktualisierung. Mit der organisatorischen REORG-Funktion kann die Schlüsselaktualisierung jederzeit nachgeholt werden. Eine weitere Verbesserung läßt sich durch die Verwendung des jeweils günstigsten Zugriffsverfahrens erreichen.

Performance-Optimierung

Um quantitative Aussagen über die Zahl der Zugriffe zu den peripheren Speichern, über Auslastung der Zentraleinheit, Optimierungsverhalten des Daten-Paging usw. zu erhalten, kann jederzeit für einen frei wählbaren Beobachtungszeitraum die Auslastungsstatistik abgerufen werden. Damit ist es möglich, Engpässe zu erkennen und ein gegebenes System gezielt zu verbessern.

Datenschutz

Schutzfunktionen sind unerläßlich, um sicherzustellen, daß systemberechtigte Benutzer nur die ihnen zugeordneten Daten mit festgelegten Zugriffen bearbeiten können. Um dieser Forderung gerecht zu werden, verfügt DVS 300 über ein feinstufiges Schutzkonzept. Die zulässige Transferart kann jedem File zusätzlich zum Benutzerkennzeichen zugewiesen werden. Ein weiterer Schutz ist durch Paßwörter gegeben. Soll hingegen verhindert werden, daß mehrere berechtigte Benutzer zur gleichen Zeit auf den gleichen Datensatz oder File zugreifen, so kann in DVS 300 die exklusive oder kurzfristige Datensatz- oder File-Belegung eingesetzt werden.

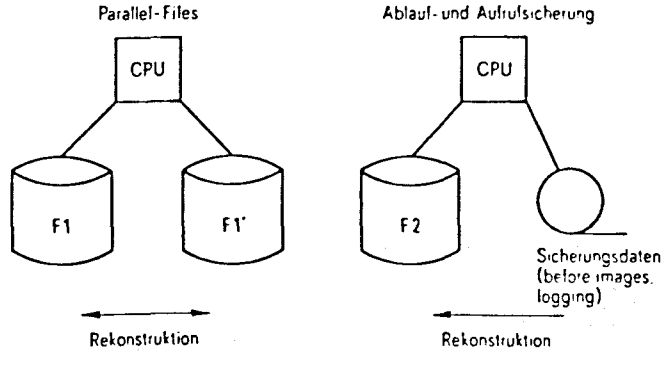
Datensicherung

Die Erfahrungen beim Einsatz von Datenverwaltungssystemen haben gezeigt, daß die Sicherung von Datenbeständen eine unverzichtbare Systemforderung ist. Von modernen Datenverwaltungssystemen wird daher erwartet, daß sie Mittel anbieten, die einen konsistenten Datenbestand bei Gerätestörungen und Software-Ausfällen gewährleisten. Diese Aufgabe übernehmen im System DVS 300 drei Sicherungsverfahren, die unabhängig voneinander oder auch einander ergänzend eingesetzt werden können.

Im Bild 5 ist die Datensicherung vereinfacht dargestellt.

Datensicherung in Parallelfiles

Damit bei Gerätefehlern kein Datenverlust auftritt, bietet DVS 300 die Möglichkeit, von einem oder mehreren Files



CPU central processing unit F File

Bild 5 Datensicherungsverfahren

auf einem weiteren gleichartigen Datenträger eine Kopie mit gleichem Namen zu führen. Schreibtransfers für Original und Kopie laufen simultan ab, Lesetransfers hingegen nur einfach. Aufgrund der Geräteüberwachung ist DVS 300 in der Lage, bei Ausfall eines Geräts mit dem verbleibenden Gerät vollwertig weiterzuarbeiten. Nachdem das Parallelgerät wieder einsatzbereit gemeldet wird, kann die Kopie on-line und parallel zu laufenden Arbeiten wieder auf den Stand gebracht werden, den das Original inzwischen hat.

Die Datensicherung in Parallelfiles bietet einen umfassenden Schutz gegen Gerätefehler. Informationsverfälschungen durch Programmabbrüche können jedoch nicht verhindert werden. Die Lösung dieses Problems mit einem jeweils unterschiedlichen Ausgangspunkt übernehmen im Rahmen von DVS 300 die Ablauf- und die Aufrufsicherung.

Ablaufsicherung

Beim Einrichten von DVS-300-Files kann festgelegt werden, ob aktuelle Informationen vor ihrer Veränderung auf einem Sicherungsdatenträger zu retten sind (before images). DVS 300 ermöglicht mit Hilfe dieser Sicherungsdaten das Wiederherstellen des ursprünglichen Datenbestands, wie er wahlweise vor dem Eröffnen oder vor der letzten Transaktion vorgelegen hat.

Die Ablaufsicherung geht davon aus, daß die wiederherzustellenden Datenträger noch lesbar sind. Um diese Lesbarkeit im System DVS 300 zu gewährleisten, kann die Funktion der Aufrufsicherung verwendet werden.

Aufrufsicherung

Im Gegensatz zur Ablaufsicherung, bei der Daten vor ihrer Veränderung gesichert werden, werden bei der Aufrufsicherung des DVS 300 von einem wählbaren Zeitpunkt an (Kopiezeitpunkt) alle datenändernden Aufrufe samt den Daten gesichert (logging). Im Störungsfall kann aus der Kopie und den Sicherungsdaten der Aufrufsicherung wieder ein aktueller Stand erzeugt werden. Neben dem Magnetband können auch Direktzugriffsspeicher als Sicherungsdatenträger verwendet werden.

Anwenderschnittstelle

Je nach Anwendungsbereich der Siemens-Systeme 300 werden unterschiedliche Programmiersprachen bevorzugt. Im Bereich der Lagerhaltung, Auftragsverwaltung usw. ist dies vornehmlich COBOL; bei technisch-wissenschaftlichen Aufgaben eignet sich besonders die Programmiersprache FORTRAN oder die problemorientierte Dialogsprache BASIC; bei Prozeßaufgaben ist hingegen der Einsatz der Assembler-Sprache noch üblich.

Sprachanschlüsse

Aufgrund dieser aufgabenorientierten Sprachbevorzugung sind die DVS-300-Aufrufe in die jeweiligen Programmiersprachen der Siemens-Systeme 300 integriert. Der COBOL-74-Anwender kann die hier angebotenen Sprachmittel direkt zum Ansprechen von DVS-300-Funktionen verwenden. Darüber hinausgehende DVS-300-Funktionen sind über eine CALL-Schnittstelle verfügbar. Der Sprachanschluß in FORTRAN 77 und BASIC erfolgt grundsätzlich über eine einheitlich aufgebaute CALL-Schnittstelle, während für Assembler-Programme Makroaufrufe zur Verfügung stehen.

Koordinierung der Benutzerzugriffe

Greifen mehrere Benutzer mit dem Ziel einer Veränderung auf den gleichen Datenbestand zu, dann müssen Koordinierungsmittel verfügbar sein, die Inkonsistenzen vermeiden. Die Koordinierungsmittel des Systems DVS 300 stehen sowohl auf der File- als auch auf der Datensatzebene zur Verfügung und räumen einem Benutzer eine exklusive File- oder Datensatzbearbeitung ein. Im Gegensatz zur exklusiven File-Belegung können bei ausschließlicher Anwendung der Datensatzsperre unterschiedliche Programme mit unterschiedlichen Datensätzen des gleichen Files arbeiten.

Die DVS-300-Anwenderschnittstelle ist unabhängig von den jeweils angeschlossenen Geräten. Dies ermöglicht eine Portabilität von Programmen und Daten. Die Berücksichtigung der jeweiligen Anlagenkonfiguration ist daher Aufgabe der Geräteschnittstelle.

Geräteschnittstelle

Dem System DVS 300 wird über Bedienungen die aktuelle Anlagenkonfiguration durch Zuordnung von Datenträgern zu Geräten mitgeteilt. Diese Zuordnung kann jederzeit geändert werden. Als Geräte sind nicht nur Direktzugriffsspeicher, sondern auch Magnetbandeinheiten und serielle Geräte zugelassen.

Um eine anwendungsgerechte Datenablage zu erreichen, kann selbst die Zuordnung von Daten zu Datenträgern außerhalb der Anwenderprogramme erfolgen. Ohne Programmänderung sind die Daten auf Magnetplatte, Magnetband, Floppy Disk oder Lochkarte hinterlegbar.

Schlußbemerkung

Im Vergleich zu einer Software-Lösung, die speziell für den Anwendungsfall erstellt wurde, bietet das standardisierte und zentrale Datenverwaltungssystem DVS 300 in den Siemens-Systemen 300 zahlreiche Vorteile bei der Verwaltung und Verarbeitung umfangreicher Datenmengen, wie generierbarer Funktionsumfang, Realisierung aller gängigen Zugriffsformen, Mehrfachindizierung, kurze Reaktionszeiten und Funktionen zur Datensicherung, Datenerhaltung und zum Datenschutz. Eine benutzerfreundliche und geräteunabhängige Anwenderschnittstelle gibt es für BASIC-, FORTRAN- und Assembler-Anwender. Für COBOL-Anwender stehen die in dieser Sprache vorhandenen Sprachmittel direkt zur Verfügung. DVS 300 betreibt sämtliche Datenträgertypen, z.B. Direktzugriffsspeicher, Magnetband, Floppy Disk. Ein Paket von Dienstprogrammen rundet die Systemeigenschaften von DVS 300 ab.

Peter Fischer
Siemens AG, E 364

Karlsruhe, den 2.5.1980

S P O O L 300

Spooling-System für Eingabe und Ausgabe alphanumerischer Daten.

- Einführung
- Leistungsmerkmale von SPOOL 300
  - SPOOL-Übersicht
  - Ausgabe-Spool
  - Eingabe-Spool
  - Umlauf-Spool
  - Allgemeine Funktionen
- Bestandteile von SPOOL 300
- Installieren von SPOOL 300
- Ansprechen des Spooling-Systems
- Anwendungsbeispiele
  - Spool-Aufrufe
  - Spooling mit Dienstprogrammen
  - Assemblieren mit Spool
  - Polling
- Zusammenfassung

Einführung

Spooling-Systeme sind bisher vorwiegend in der kommerziellen Datenverarbeitung üblich, weil dort hauptsächlich rechen- und transferintensive Aufgaben zu lösen sind. Transferintensiv heißt, daß ein reger Datenaustausch stattfindet zwischen Zentraleinheit, schnellen und langsamen peripheren Geräten.

S		**S**imultaneous	
P		**P**eripheral	
O	ist abgeleitet von	**O**perations	und
O		**O**n	
L		**L**ine	

ermöglicht eine zeitoptimale Ausnutzung der Datenverarbeitungsanlage.
Die Verarbeitungsgeschwindigkeit druckender oder stanzender Geräte ist um ein Vielfaches langsamer als die der Zentraleinheit. Infolgedessen muß der Zentralprozessor relativ lange in einer Untätigschleife auf die peripheren Abschlußmeldungen der Ein-/Ausgabe-Operationen warten.
Spooling-Systeme wirken dieser ineffektiven Auslastung des Zentralprozessors entgegen, indem sie die zwischen Programmen und langsamen peripheren Geräten zu transferierenden Daten auf einem Plattenspeicher (Randomspeicher) zwischenpuffern und zu einem späteren Zeitpunkt weiterleiten. Damit entkoppeln sie die Laufzeit der aufrufgebenden Programme vom zeitlichen Verhalten der Geräte und ermöglichen eine zeitoptimale Auslastung der Zentraleinheit mit schneller und langsamer Peripherie.
Die Rechner der Siemens Systeme 300 dienen, neben ihrer Verwendung als reine Prozeßrechner, in zunehmendem Maße auch der Softwareerstellung, sowie dem Austausch alphanumerischer Daten parallel zum Prozeßbetrieb.
Daraus resultierte zwangsläufig die Forderung, für die Peripherspeicherbetriebssysteme ORG 300 P und PV der Mini-Computer R10, R20, R30 und R40 ein Spooling-System zur Verfügung zu stellen.
SPOOL 300 ist ein Spooling-System für Eingabe und Ausgabe alphanumerischer Daten.
Als Vorläufer von SPOOL 300 kann die unter dem Namen „Datenpufferung" bekannte Funktion der Peripherspeicherbetriebssysteme betrachtet werden.
Die Funktion Datenpufferung ist generierbar. Sie übernimmt Ein-/Ausgabe-Aufrufe der Programme an langsame Geräte, puffert die Aufrufdaten auf einem Randomspeicher zwischen und veranlaßt - stellvertretend für die aufrufgebenden Programme - die Ein-/Ausgabe-Operationen nach Maßgabe der Geräteverfügbarkeit.
Diese Betriebssystemfunktion entkoppelt - ebenso wie SPOOL 300 - die Programmlaufzeiten von der Verarbeitungsgeschwindigkeit langsamer Geräte, sie reduziert die Verweilzeiten peripherspeicherresidenter Programme im Laufbereich und trägt damit auch zu einer effektiveren Anlagenausnutzung bei.
Bei SPOOL 300 werden die Daten auf einem Plattenspeicher programm- und gerätespezifisch zwischengepuffert. SPOOL 300 realisiert die Belegfunktion für serielle Geräte. Damit ist sichergestellt, daß die Texte bei Ausgabe auf das jeweilige „reale Gerät" zusammenhängend ausgegeben werden.

Leistungsmerkmale von SPOOL 300

SPOOL-Übersicht

Das Spool-System kann von Anwender- oder von Systemprogrammen über alphanumerische Standard-Ein/Ausgabe-Aufrufe $ STEIAL bzw. $ STAUAL für virtuelle Geräte angesprochen werden. Virtuelle Geräte werden per Aufruf oder Bedienung definiert, sie dürfen weder generiert noch installiert sein. Mit der Angabe eines virtuellen Gerätes ist die Eigenschaft „SPOOLING" zwingend verbunden.
Der Eingabe-Spool liest die alphanumerischen Daten langsamer serieller Eingabegeräte und sammelt sie in der SPOOL-Datei auf einem Direktzugriffsspeicher (PSD).
Der Einlesevorgang wird durch Kommando oder zyklisch angestoßen, sobald das Eingabegerät sendebereit ist.
Zu einem beliebigen späteren Zeitpunkt können die in der SPOOL-Datei gesammelten Daten von dem Empfängerprogramm abgerufen werden.
Das Ausspool-System puffert alphanumerische Daten von Programmen an langsame Geräte in der SPOOL-Datei und gibt sie zeitentkoppelt von diesen Programmen an die peripheren Geräte aus. Der Ausgabe-Anstoß erfolgt per Kommando oder automatisch bei Datenstromende oder Programmende.
Die SPOOL-Datei ist eine vom Systemgenerator zusammenhängend eingerichtete Datei zum Abspeichern der Spooldaten.
Die Funktion SPOOL-Auskunft gibt dem Benutzer einen Überblick über die existierenden Spools und deren Status.
Spools sind mit alphanumerischen Texten gefüllte Datenbereiche innerhalb der Spool-Datei, die nur mit Ausgabe- oder nur mit Eingabeaufrufen an virtuelle Geräte angesprochen werden können. Mehrere Spools können zu einer Spoolgruppe zusammengefaßt werden.
Wir unterscheiden Ausgabe- und Eingabe-Spools.
Zunächst sollen die Leistungsmerkmale des Ausgabe-Spools betrachtet werden.

Ausgabe-Spool

Die gespoolten Texte werden auf Externspeicher abgelegt und können von dort programmiert oder über Bedienung abgerufen werden. Außerdem besteht die Möglichkeit der automatischen Ausgabe bei Programmende.
Die Ausgabe kann mehrfach erfolgen. Die Daten können nach Ausgabe wahlweise gelöscht oder gesichert werden. Bei Sicherung ist eine beliebige Anzahl von Ausgabewiederholungen möglich. Bei Ausgabe mit höchster Priorität kann auf 2 Geräten parallel ausgegeben werden.
Die Ausgabe eines Spools auf ein Realgerät ist unterbrechbar. Sie kann vom Operator an der Unterbrechungsstelle, an einem „Checkpoint" oder über eine „Displayfunktion" an einer beliebigen Stelle fortgesetzt, bzw. neu begonnen werden.
Die Ausgabe der Spool-Gruppen läuft prioritätsgesteuert ab.
Je niedriger eine Spoolgruppennummer, um so höher ist ihre Priorität. Durch den Operator kann fallweise eine Ausgabe mit höchster Priorität erfolgen.

Vor der eigentlichen Ausgabe kann zusätzlich eine Meldung an das Standard-Meldegerät ausgegeben werden, wodurch zusätzliche Maßnahmen vom Operator veranlaßbar sind (z. B. spezielles Format einlegen).
Der Name von virtuellen Geräten gleicht formal dem der Realgeräte. Die Gerätekennung ist frei wählbar (z. B. ABCD oder OTTO), allerdings darf ein virtueller Gerätename nicht mit dem Namen eines Realgerätes übereinstimmen.
Es können Probedrucke vorgenommen werden, wobei hier anstelle von abdruckbaren Zeichen das Zeichen „X" ausgegeben wird. Hierbei kann z. B. eine Formulareinstellung eines Druckers vorgenommen werden, ohne daß der Einsteller Kenntnis über den Ausgabetext erhält.
Einer Spoolgruppe können bis zu 8 reale Ausgabegeräte zugeordnet werden. Die Ausgabe erfolgt dann auf einem freien Gerät dieser Zuordnungsliste. Als Ausgabegeräte sind alle realen (lokale und globale) Geräte zugelassen, die alphanumerische Texte verarbeiten. Eine Codeverträglichkeit mit dem Ausgabetext wird vom Anwender gefordert.

Nicht mehr benötigter Pufferplatz (abgearbeitete Spools) kann dem SPOOL-System durch Löschen der Spools wieder zur Verfügung gestellt werden. Spools sind löschbar, auch wenn sie noch keine Daten enthalten.

Es können so lange Ausgaben in den Spool erfolgen, wie Platz in der SPOOL-Datei vorhanden ist. Die Größe dieser Datei wird beim Generieren festgelegt. Die maximale Länge eines Spools ist damit nur durch die Länge dieser Datei begrenzt.

Anwenderprogramme können die Ausgabe eines Spools auf ein reales Gerät auch in eigener Initiative übernehmen. Hierfür ist eine Schnittstelle vorhanden (Koordinierungszähler).

Die Ausgabe-Reihenfolge der Spools erfolgt nach Priorität, d.h. Spools mit niedriger Spoolgruppennummer werden mit hoher Priorität ausgegeben.

Die einer Spoolgruppe zugeordneten Realgeräte (bis zu 8 je Spoolgruppe) können zu Gruppen zusammengefaßt werden. Die Ausgabe erfolgt dann auf alle Geräte einer Gruppe parallel, wobei das langsamste Gerät die Ausgabegeschwindigkeit bestimmt. Für den Ausgabebeginn ist die gleichzeitige Ausgabebereitschaft aller Geräte der Gruppe notwendig.

Pufferplatz für in Abarbeitung befindliche Spools kann mit der Abarbeitung mitlaufend freigegeben, d. h. dem System wieder zur Verfügung gestellt werden.

Eingabe-Spool

Zum Einspoolen zugelassen sind alle realen (lokale und globale) Geräte, die alphanumerische Eingaben ermöglichen, z. B. LKAE, LSTE.

Die realen Eingabegeräte können vom Einleseprogramm wahlweise zyklisch automatisch oder kommandogesteuert zum Einlesen eines Eingabe-Spools angestoßen werden.

Eingabe-Spools werden durch alphanumerische Eingabeaufrufe (STEIAL) an virtuelle Geräte satzweise übernommen. Am Spoolende wird der Aufruf mit Anzeigen abgewiesen.

Grundsätzlich besteht die Möglichkeit, durch Ausgabe-Aufrufe an virtuelle Geräte ausgegebene Information durch Eingabe-Aufrufe an das gleiche Gerät wieder einzulesen. Dies kann auch von einem anderen Programm aus erfolgen.

Die eingespoolten Daten können auch mehrfach von Programmen eingelesen werden. Sie werden erst durch ein explizites Löschkommando gelöscht.

Umlauf-Spool

Zum gleichzeitigen Betrieb eines Spools für Ein- und Ausgabe wird ein Verfahren zur Verfügung gestellt, bei dem die gespoolten Datensätze in einem Umlaufpuffer gespeichert werden, d.h. abhängig von Speicherplatz und Datenmenge werden „alte" Informationen überschrieben und sind damit nicht mehr zugänglich. Während für Ein/Ausgabe-Spools der benötigte Platz von der anfallenden Datenmenge abhängt, wird beim Umlauf-Spool die Größe des Umlaufpuffers laufzeitkonstant durch Aufruf oder Bedienung vorgegeben.

Für den Umlauf-Spool werden getrennte Schreib/Lese-Zeiger geführt. Der Lesezeiger kann hierbei den Schreibzeiger nicht überholen. Wurde für Schreib- oder Lesezeiger ein Checkpoint gesetzt, so kann der jeweils andere Zeiger diesen Checkpoint nicht überschreiten. Checkpoint setzen heißt, der aktuelle Schreib- oder Lesezeiger eines Spools wurde in den Merkzellen des Spoolbuchhalters hinterlegt.
Erreicht der Schreibzeiger den Lesezeiger, so werden nachfolgende Ausgabeaufrufe entweder abgewiesen oder, bei entsprechend definiertem Umlauf-Spool, der Lesezeiger mit dem Schreibzeiger mitgeführt. Dies bedeutet, daß ab diesem Zeitpunkt die überschriebene Information verloren ist. Diese Funktion kann z. B. bei Fehlermeldungsausgabe oder Testmeldungen ausgenutzt werden.

Ein Umlauf-Spool kann wie ein Ausgabe-Spool durch das Spool-System auf ein Realgerät ausgegeben werden.

Allgemeine Funktionen

Es besteht die Möglichkeit, zu einem beliebigen Zeitpunkt den aktuellen Zeiger-Stand (Schreib- oder Lesezeiger) des Spools zu merken. Bei Betrieb als Umlaufpuffer können beide Zeigerstände gemerkt werden. Gegebenenfalls (z. B. Protokollgerät defekt) ist dann die Ein- oder Ausgabe ab diesem vorher gemerkten Stand wiederholbar.

Einer Spoolgruppe können zu jedem Zeitpunkt bis zu 8 Realgeräte zugeordnet werden, d.h. auch dann, wenn noch kein Spool für diese Spoolgruppe existiert.

Die Anwenderprogramme können sich vom Spoolsystem Informationen über den aktuellen Zustand der Spools geben lassen.

Nicht in Ausgabe befindliche Spools können auf den Spoolanfang oder auf einen Checkpoint (soweit definiert) rückgesetzt werden. Bei Umlaufpufferbetrieb wird beim Rücksetzen des Schreib- oder Lesezeigers auf Spoolanfang auf den jeweils anderen Zeiger rückgesetzt.

Der Anwender kann bei nicht in Ausgabe befindlichen Spools den Spoolinhalt durch seitenweise Ausgabe auf ein Terminal überprüfen, bzw. eine bestimmte Stelle suchen. Diese Funktion ist nur als Kommandofunktion realisiert.

Löschaufrufe, die für nicht löschbare Spools (z. B. in Ausgabe befindliche) abgegeben werden, bleiben solange gespeichert, bis der betreffende Spool in einen Zustand gelangt, der das Löschen zuläßt.

Der Anwender kann Realgeräte temporär für die Bearbeitung durch das Spoolsystem sperren, um das Gerät in eigener Regie zu verwalten. Eine während des Sperraufrufes tätige Ausgabe des Spoolsystems wird nicht unterbrochen.

Bestandteile von SPOOL 300

Der gesamte Funktionsumfang von SPOOL 300 ist teilweise in ORG-Bausteinen und teilweise im SPOOL-Programmsystem realisiert. Die ORG-Bausteine verwalten die SPOOL-Datei und puffern die eintreffenden Datensätze in den zugeordneten Spool.
Das Spoolprogrammsystem gliedert sich in die Programme SPOOL, SPOOLP, SPOOLF und SPPOLL.
Das Bedienprogramm SPOOL analysiert die Kommandos und gibt die ORG-Aufrufe ab. SPOOL ist monitorfähig. SPOOLP wickelt die Ausgabe ab, während der Pufferfüllteil SPOOLF abgeschlossene Ausgabe-Spools automatisch oder kommandogesteuert ausgibt. SPOOLP und SPOOLF sind nicht bedienbar und werden stets nur von SPOOL aktiviert. SPPOLL realisiert die Polling-Funktion, d. h. das automatische Einlesen von Datensätzen, sobald das Eingabegerät sendebereit ist.
SPPOLL muß für jedes zu pollende Gerät geladen werden.

Installieren von SPOOL 300

Beim Installieren des SPOOL-Systems unterscheiden wir zwischen einer Generierphase und einer Ladephase.
Aufgrund der Generierparameter /W:ASPL; und /DL: ....
setzt der Systemgenerator die ORG-Bausteine im HSP bzw. auf PSD ab.
Gleichzeitig legt er die sog. virtuelle Geräteliste im HSP sowie die SPOOL-Datei auf PSD an.
Die SPOOL-Datei ist zusammenhängend eingerichtet, sie kommt im gesamten System nur einmal vor und ist nicht verlängerbar.
In der Ladephase sind die Programme SPOOLP, SPOOLF und SPOOL zu laden sowie SPPOLL, falls die Pollingfunktion genutzt werden soll.
SPOOLP ist zwingend hsp-resident zu laden, während SPOOLF, SPOOL und SPPOLL wahlweise als HRP oder als PRP ladbar sind.

Ansprechen des Spooling-Systems

Die zum Ansprechen des Spooling Systems notwendigen Standard-E/A-Aufrufe können vom

- SPOOL-Bedienprogramm, von
- Anwenderprogrammen oder vom
- Polling-Programm

abgegeben werden.

Das Kommando SPLOAD wird dabei SPOOL-intern aufgelöst in einen Eingabeaufruf für reale und einen Ausgabeaufruf für virtuelle Geräte: d.h., die alphanumerischen Daten werden zunächst vom physikalischen Eingabegerät gelesen und anschließend in einen Spool auf PSD abgespeichert.
Bei Standardausgabe-Aufrufen aus Anwenderprogrammen für reale Geräte erfolgt normale Bearbeitung, d.h. die alphanumerischen Daten werden unmittelbar auf das reale Gerät ausgegeben.
Ist im zugehörigen Geräte-Daten-Block (GEDA-Block) jedoch ein virtuelles Gerät angegeben, so erfolgt Zwischenpufferung der Daten in der Spool-Datei .
Das Polling-Programm fragt nach Geräte-Zuteilung das zu pollende Eingabe-Gerät alle 10 sec. ab. Dabei wird zunächst ein $ STEIAL auf das reale Gerät und anschließend ein $ STAUAL auf das zugeordnete virtuelle Gerät abgesetzt (Steuerkarte /VD: virtger, parlist;).
Bei betriebsbereitem Gerät werden die alphanumerischen Daten eingelesen, wobei als erstes Kartenäquivalent eine Steuerkarte mit Angabe des zugeorndeten virtuellen Gerätes erwartet wird.
Ist das Gerät unklar, so wird nach jeweils 10 sec. erneut abgefragt. Der Einleseversuch wird bei unklarem Gerät nach 2 Minuten abgebrochen.
Wird das Gerät während des Einlesevorgangs unklar, so erfolgt eine Meldung auf dem Standardmeldegerät. Nach Klarschalten des Gerätes wird der Einlesevorgang selbständig fortgesetzt.

Das Ausspoolen aus der SPOOL-Datei kann vom

- SPOOL-Bedienprogramm bzw. von
- Anwenderprogrammen

initiiert werden.
Dazu stehen die Kommandos SPOUT/SPOUTI (Prioritätsgesteuertes Abrufen/Abrufen mit höchster Priorität) bzw. die Aufrufe $ STEIAL für virtuelle Geräte zur Verfügung.

Dem eigentlichen Ausspoolvorgang kann ein Probedruck vorausgehen, wobei alle Zeichen ungleich „Blank" im Druckbild mit X erscheinen.
Parallelausgabe auf 2 reale Geräte ist ebenso möglich, wie Anhalten und Fortsetzen.

Anwendungsbeispiele

- Spool-Aufrufe
- Spooling mit Dienstprogrammen
- Assemblieren mit Spool
- Polling

Zusammenfassung

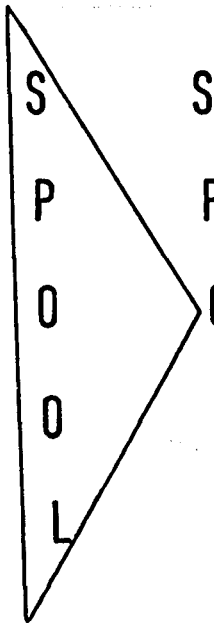
Mit SPOOL 300 – einem Subsystem von ORG 300 P und PV – steht dem Anwender ein leistungsfähiges Instrument zur optimaleren Ausnutzung seiner Prozeßrechneranlage zur Verfügung.

SPOOL 300
- entkoppelt die Programmlaufzeiten von der Verarbeitungsgeschwindigkeit langsamer serieller Geräte durch Zwischenpufferung der zu transferierenden alphanumerischen Daten auf einem schnellen Speichermedium und
- erhöht gleichzeitig die Verfügbarkeit peripherer Geräte durch die Möglichkeit, die gesammelten Daten zu einem beliebigen späteren Zeitpunkt dem Empfänger zuzuleiten.

Beispielsweise können die im Closed-shop-Betrieb des Rechenzentrums erzeugten Übersetzungsprotokolle von Programmen in der Spooldatei gesammelt und in Zeiten geringerer Anlagenauslastung (z. B. während der Nachtstunden) auf Schnelldrucker ausgegeben werden.

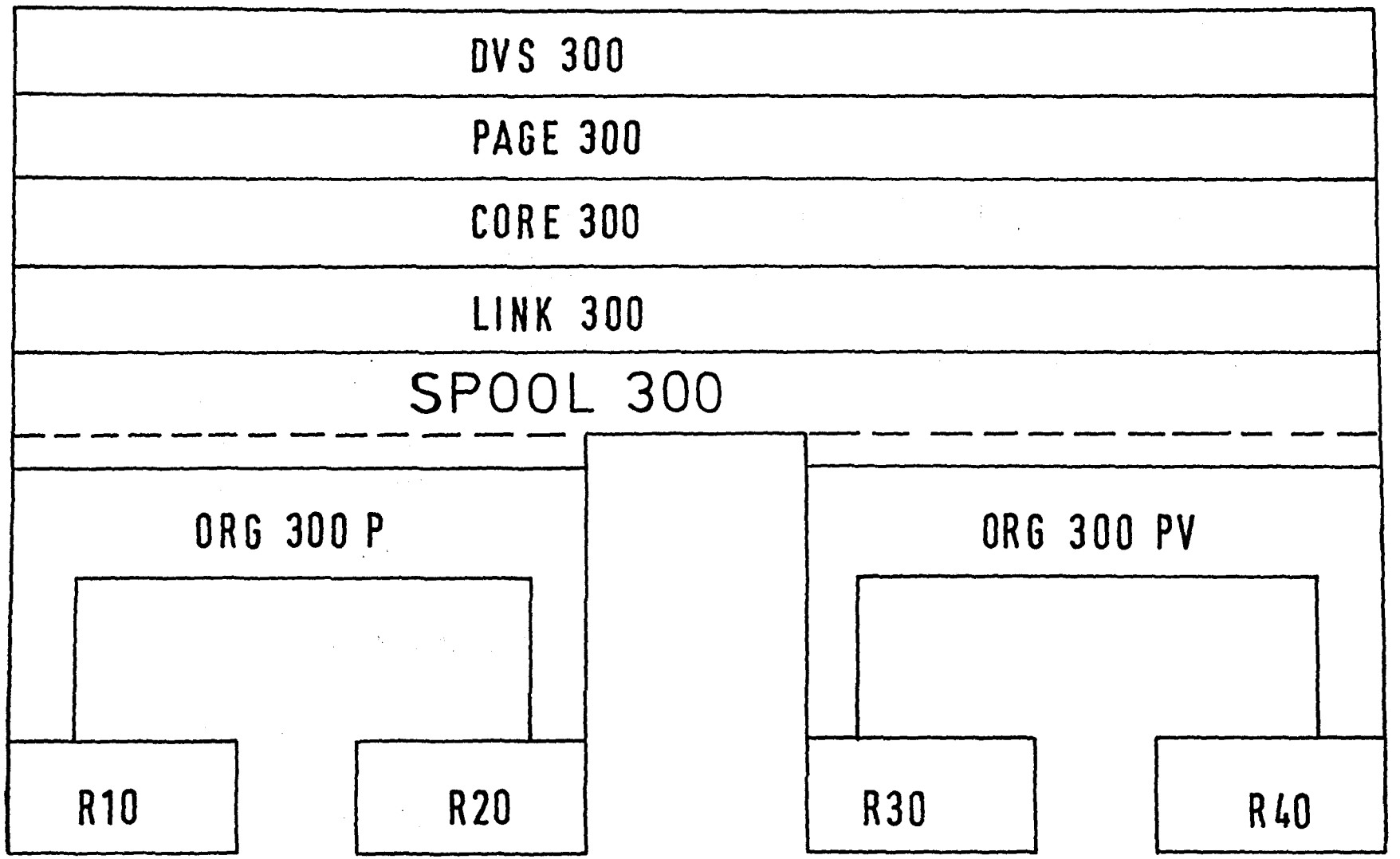
Allgemeiner Hinweis:

Die auf den Folien mit + gekennzeichneten Leistungsmerkmale von SPOOL 300 stehen erst mit Ausgabe C-A1 des Produkts zur Verfügung.



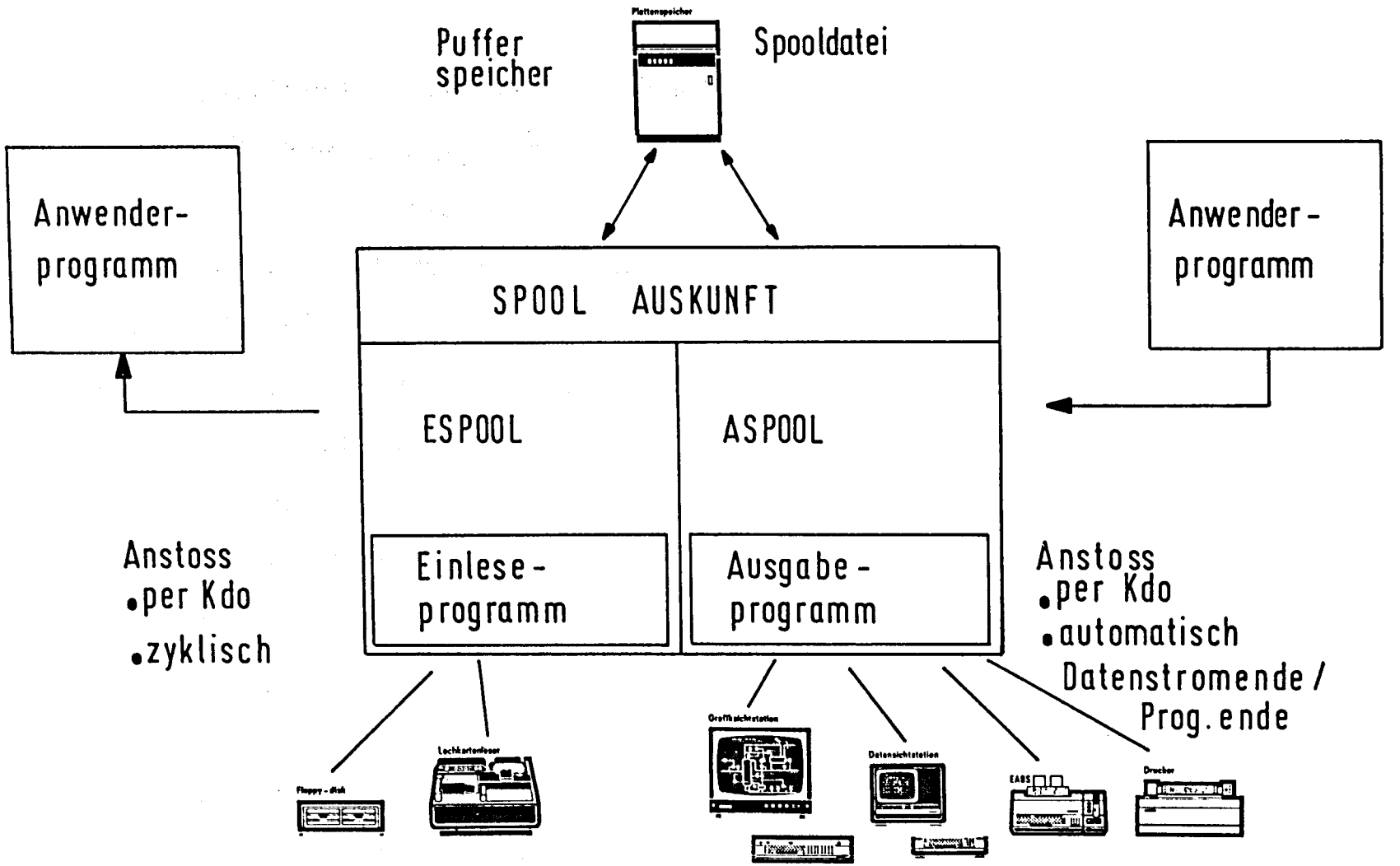
Simultaneous
Peripheral
Operations
On
Line

SPOOL 300:
- Spooling-System für Eingabe und Ausgabe
- Zeitoptimale Ausnutzung der ZE mit schneller und langsamer Peripherie



SPOOL 300 mit Systemumgebung

E 364
4/80



SPOOL-System

E 364
4/80

- Zwischenspeicherung ,Abruf
- Mehrfache Ausgabe
- Unterbrechbarkeit der Ausgabe
- Ausgabepriorität
- Zusatzmeldung
- Virtuelle Gerätenamen
- Probedruck
- Zuordnung der Ausgabegeräte
+ — Freigabe abgearbeiteter Spools
- Spoollänge
- Ausgabe durch Anwenderprogramme
+ — Ausgabereihenfolge
+ — Allgemeine Parallelausgabe
+ — Mitlaufendes Freigeben von Spools

99

- Zugelassene Geräte
- Eingabeanstoß
- Übernahme der gespoolten Daten
- Eingabe ausgespoolter Daten
- Mehrfache Eingabe

- Umlauf-Spool
- Getrennter Schreib/Lese-Zeiger
+ — Lesezeiger als Schleppzeiger
+ — Ausgabe von Umlauf-Spools

— Checkpoint

\+ — Zuordnung Spoolgruppe-Realgerät

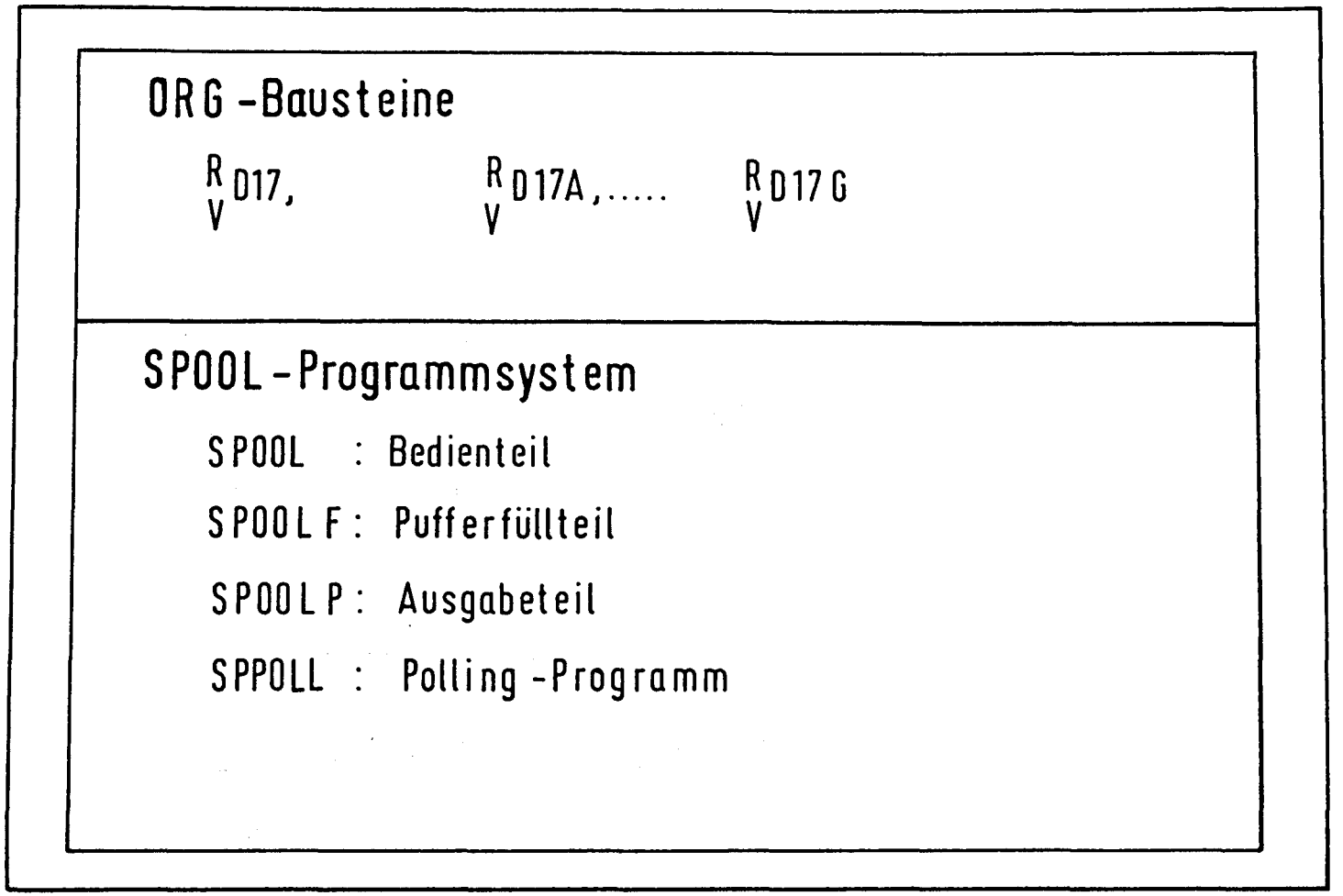
\+ — Spoolauskunft an Anwenderprogramme

— Rücksetzen

\+ — Display-Funktion

\+ — Speicherung von Löschaufrufen

\+ — Sperren von Realgeräten



Bestandteile von SPOOL 300

Generierphase

/W:ASPL,/DL:logpsd,lan,anzger; ORG-Masterstapel

SYSTEMGENERATOR

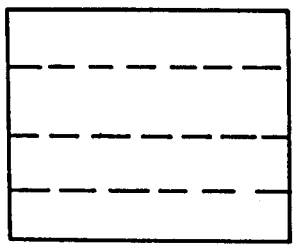
HSP

- ORG-Bausteine
- GERÄTELISTE

für reale Geräte
für virtuelle Geräte

PSD

- ORG-Bausteine
- SPOOL-Datei: YXs SPO ORG PV
  YXSPO ORG P
- Zusammenhängend
- Nur 1x im System
- Nicht verlängerbar



Ladephase

/LADE:<..,.>,SPOOLP...:P nr, +)

/LADE:<..,.>,SPOOLF...:{P nr; ++)
SPOOL ...:{PP nr;

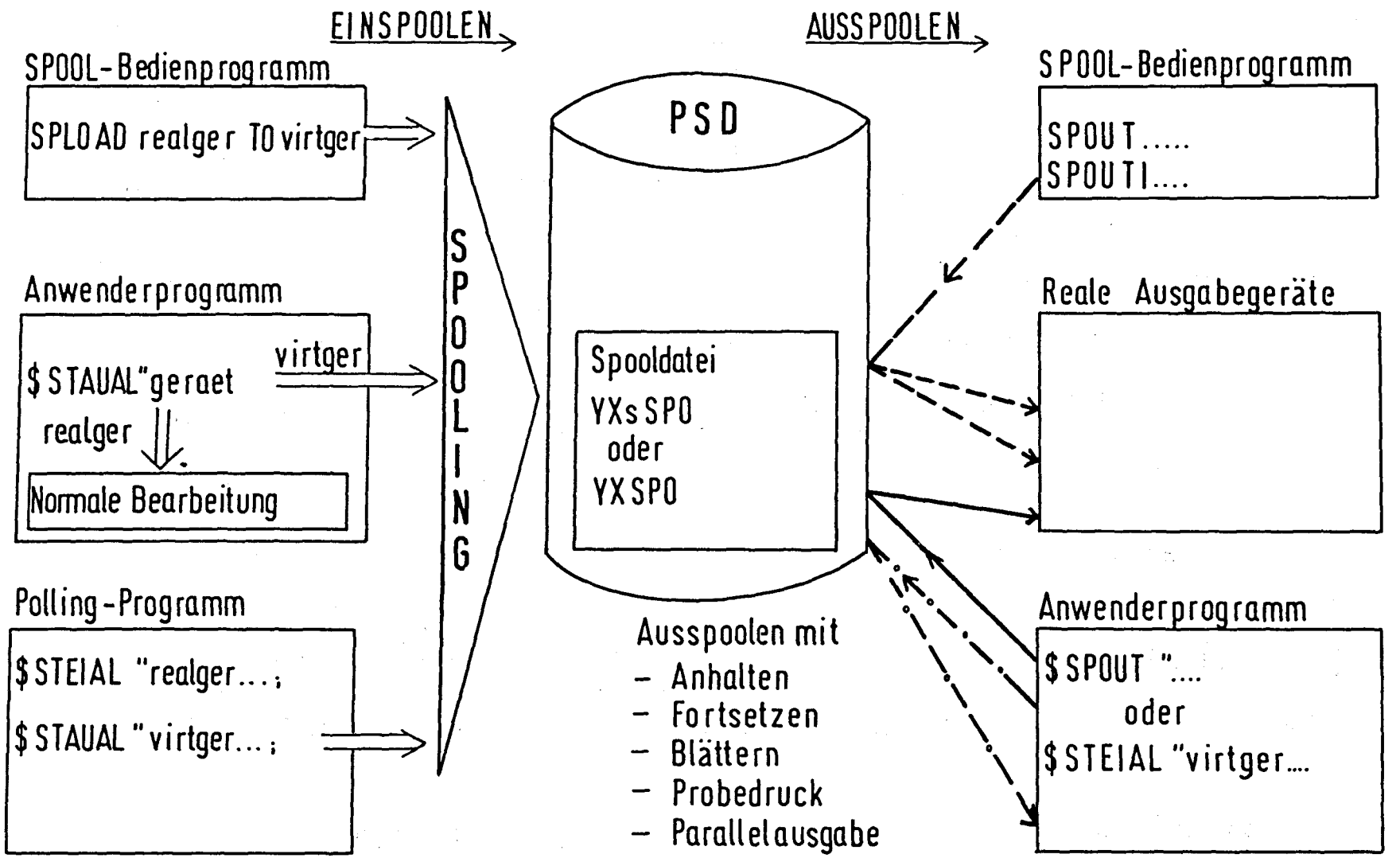
/LADE:<.,.>,SPPOLL...:{P nr; +++)
{PP nr;

+) zwingend HRP!!

++) HRP oder PRP

+++) SPPOLL mehrfach ladbar
1x pro zu pollendes Gerät

/STRT:SPOOL;



SPOOL 300 : Ansprechen des Spooling

E 364
4/80

```
'NAME'    AUFRUF
'SATZ'    SPOOL BEISPIEL

'VA'      $ RUFORG" VIRTDEF"T@V;
'VA'      $ RUFORG"ASSDV  " T@V;
'VA'      $ RUFORG" STAU  "  T@V;
          :
'VA'      $ RUFORG"ENDE "   T@V;
```

```
'VA' VIRTDEF/$ CRDV"GEDA0D""A"S"R@2;
```

- Eröffnen Ausgabespool
- Buchführung anlegen im HSP und in der SPOOL-Datei
  Eigenschaften des SPOOLS :
  - –ORDINARY
  - –AUTOMATIC
  - –SAVE
  - –2 Duplikate

```
'VA' ASSDV/$ ASSDV"......"DG@DRUA(Ø);
'VA' STAU / $ STAUAL"GEDA0D".....;
'VA' GEDA0D/$ GEDA0D"LOG@DRUA100".....;
'VA' ENDE / $ ENDE;
```

- Zuweisen des realen Gerätes DRUA (Ø);

Ausgabespool ausgabebereit $\Longrightarrow$
automatische Ausgabe auf ein reales Gerät
bei Datenstromende

Jedes simultan ablaufende Dienstprogramm benutzt sein eigenes virtuelles Gerät ⟹ programmspezifische Ausgabe der Spools

CRDV virtger	Definition eines virtuellen Gerätes mit bestimmten Eigenschaften
ASSDV SG-nr TO DRUA,DRUA1	Zuweisung einer Spoolgruppe zu maximal 8 realen Geräten

Kommando an Dienstprogramm
} Ausführung der Funktion
evtl. weitere Kommandos

SPCLOSE virtger		Der Spool wird ausgabebereit markiert
SPOUT	virtger SG-nr. ALL	Ausgabe des/der Spools

Kommandogesteuert

```
:SPOOL:CRDV DRUA47.11 CMD SAVE

:SPOOL:SPCLOSE DRUA47.11
:SPOOL:SPOUT DRUA  47.11
```

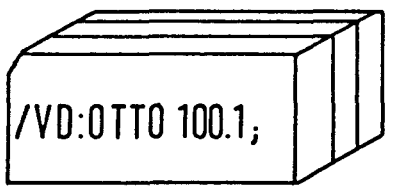
automatisch

```
:SPOOL:CRDV DRUA47.11 SAVE

:SPOOL:ASSDV SG-47 TO DRUA
:AS30B:T:<DRUA(47.11)>;
:AS30B:RUN:EQ,TEST;

:AS30B:ENDE;
```

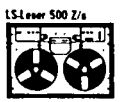
75

SPOOL-Beispiel: Assemblieren,
Kommandogesteuerte und automatische Ausgabe des Spool

```
:SPOOL:CRDV OTTO100.1
/EG:< PLSK,GSV>;
/LADE:EG,SPPOLL:LB2:PP20;
/STRT:P20;
   prnr SPPOLL:GIVE REALDEVICE!

:SPPOLL:POLL LKAE
```




```
:SPOOL:CRDV SUSI40
/EG:<PLSK,GSV>;
/LADE:EG,SPPOLL:LB1:PP21;
/STRT:P21;
   prnr SPPOLL GIVE REALDEVICE!

:SPPOLL:POLL LSTE
```



SINEC 300

Das Kommunikationssystem SINEC® 300 (Siemens-Netzwerk für Minicomputer 300) bietet eine aufeinander abgestimmte Palette von Software- und Hardware-Produkten zur Kopplung von Siemens-Systemen 300 zu beliebig konfigurierbaren homogenen Rechnernetzen. Da SINEC 300 außerdem als offenes Kommunikationssystem konzipiert ist, ermöglicht es auch den Aufbau von heterogenen Rechnernetzen. Schnittstellen zum Anschluß von Systemen 300 an öffentliche Datenpaketvermittlungsnetze (Datex P) gehören zum Standard-Funktionsumfang.

SINEC 300 enthält standardmäßig auch Programmschnittstellen zur Integration der Systeme 300 in bestehende Netzwerkarchitekturen wie IBM-SNA (SNA System Network Architecture) und TRANSDATA®. Darüber hinaus gehört zu SINEC 300 als wesentliches Leistungsmerkmal eine Protokollbeschreibungssprache PDL (PDL Protocol Description Language), mit deren Hilfe die notwendigen Anpassungen an fremde Kommunikationsprotokolle vorgenommen werden können. Dadurch kann das Spektrum der möglichen Fremdrechnerkopplungen entsprechend dem jeweiligen Anwendungsfall erweitert werden.

Leistungsumfang des offenen Kommunikationssystems SINEC 300

Mit SINEC 300 steht Anwendern der Siemens-Systeme 300 ein Kommunikationssystem zur Verfügung, mit dem sich Rechnerverbunde unabhängig von der Netzstruktur (z.B. hierarchisch, vermascht), von der geographischen Verteilung der Rechner und von der Art und Geschwindigkeit der Datenübertragungsleitungen aufbauen lassen.

Je nachdem, ob ein Rechnernetz nur auf die räumliche Verteilung der Computerleistung oder zusätzlich noch auf die netzweite Verfügbarkeit von Betriebsmitteln wie Peripheriegeräte oder Datenbestände abzielt, können mit den Systemen 300 Nachrichten-, Funktions- und Datenverbunde aufgebaut werden.

SINEC-300-*Nachrichtenverbund*

Ein Rechnernetz als Nachrichtenverbund dient der Übermittlung von Nachrichten (Daten) zwischen Anwender-, System- und Dienstprogrammen, die in verschiedenen autonomen Rechnern ablaufen. So stellt SINEC 300 in einem Nachrichtenverbund mit den Systemen 300 dem Anwender Kommunikationsaufrufe, ähnlich den gewohnten Zugriffsoperationen zu Dateien, zur Verfügung. Die Daten für die Programme werden unabhängig von den speziellen Charakteristiken der Übertragungsleitungen und den verwendeten Datenkommunikationsprotokollen übertragen.

Die Kommunikationsaufrufe von SINEC 300 sind in der Assemblersprache der Systeme 300 und den höheren Programmiersprachen FORTRAN und COBOL als CALL-Aufrufe enthalten.

Kopplungen zu Fremdrechnern

Nachrichtenverbunde können mit SINEC für homogene Rechnernetze, d.h. Netze, in denen ausschließlich Rechner der eigenen Systemfamilie gekoppelt sind, und für heterogene Rechnernetze, d.h. Netze mit Kopplungen zu Fremdrechnern, realisiert werden (Bild 1). Hier unterstützt das *offene* Kommunikationssystem SINEC 300 standardmäßig die Anpassung der SINEC-Kommunikationsprotokolle an die äquivalenten Kommunikationsprotokolle der Fremdsysteme.

Im heterogenen (offenen) Rechnerverbund oder bei einfachen Kopplungen zu Fremdsystemen ist es notwendig, die Protokolle aller an die Systeme 300 angekoppelten Rechner zu vereinbaren. Im einfachsten Fall benutzen die mit SINEC gekoppelten fremden Kommunikationssysteme auch die SINEC-internen Kommunikationsprotokolle. Für die Fälle, in denen das nicht möglich ist, übernimmt ein Protokollumsetzer in SINEC die Anpassung der SINEC-Kommunikationsprotokolle an die der Fremdrechner.

Um den Anwendern oder einem Rechnernetzbetreiber bei Kopplungen von Systemen 300 zu Fremdrechnern die Formulierung der Protokollumsetzung zu erleichtern, stellt SINEC die Sprache PDL zur Verfügung. Die Sprachmittel der SINEC-PDL sind an die besonderen Erfordernisse der Kommunikationsprotokolle angepaßt.

Kopplungen zu IBM-Datenverarbeitungsanlagen

Speziell für den Anschluß von Systemen 300 an Datenverarbeitungsanlagen (DVA) von IBM, z.B. IBM /370 oder IBM /303x, bietet SINEC dem Anwender verschiedene Kopplungsmöglichkeiten. Der Anschluß an die IBM-DVA erfolgt dabei über SDLC-Punkt-zu-Punkt- (SDLC Synchronous Data Link Communication) oder -Mehrpunktleitungen an einen Communications Controller IBM /370x. Es können sowohl einzelne Systeme 300 als auch SINEC-Rechnernetze an eine IBM-DVA gekoppelt werden.

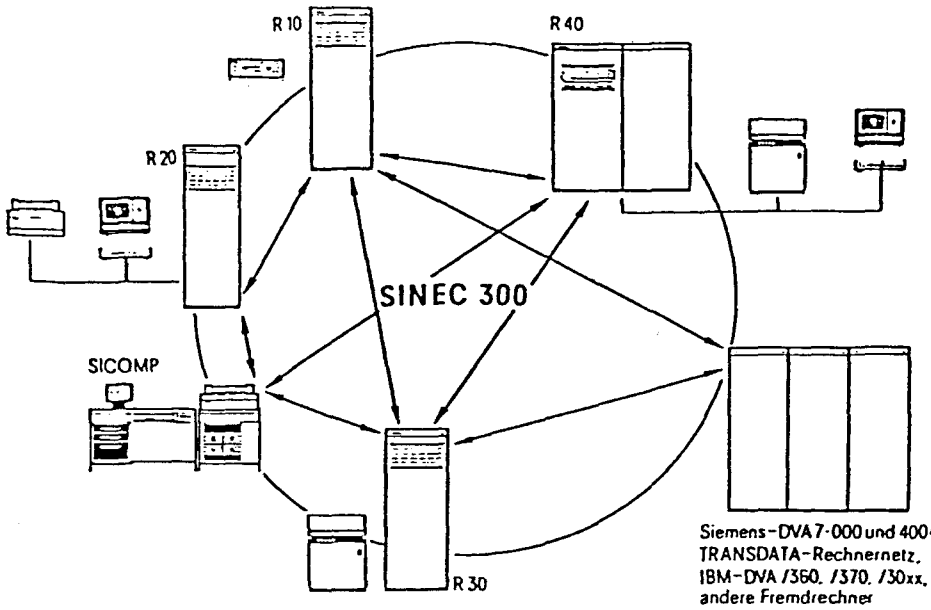


Bild 1
SINEC 300 zur Rechnerkommunikation innerhalb des Systems 300 und mit Fremdrechnern

Gegenüber der IBM-DVA verhält sich ein System 300 wie ein über SDLC-Leitungen angeschlossener SNA Cluster Controller und ermöglicht damit dem Anwender über ein SNA-Netz die Zusammenarbeit mit Programmsystemen wie beispielsweise POWER/VS, RES, JES2, JES3 sowie IMS, CICS.

Kopplungen zu Siemens-Datenverarbeitungsanlagen

Ein Schwerpunkt für die Kopplung der Systeme 300 mit SINEC ist der Anschluß an die Siemens-DVA 7·700, 7·500, 4004 und 7·800 mit den Betriebssystemen BS1000, BS2000 und BS3000 sowie an TRANSDATA-Kommunikationsrechner (s. Bild 1).

Für Kopplungen der Systeme 300 zu DVA 7·700 und 7·500 mit dem Betriebssystem BS2000 ermöglicht SINEC den Datenaustausch im Teilhaberbetrieb (DCAM), im Teilnehmerbetrieb (TIAM) und im Fernstapelbetrieb (RBAM). Bei Kopplungen zu Siemens-DVA mit dem Betriebssystem BS1000 wird der Teilhaberbetrieb (TCS) und der Fernstapelbetrieb (RSPOOL) unterstützt.

Je nach Anwendungsfall besteht die Möglichkeit, die Systeme 300 über schnelle Nahkopplungen an den Byte-Multiplexkanal einer Siemens-DVA 7·700, 7·500 oder 4004 oder über Fernsprech- oder Datexleitungen an TRANSDATA-Kommunikationsrechner zu koppeln. Es können sowohl einzelne Systeme 300 als auch SINEC-Rechnernetze mit einer Siemens-DVA gekoppelt werden. Gegenüber der Siemens-DVA verhält sich ein System 300 wie ein TRANSDATA-Kommunikationsrechner oder eine andere Siemens-DVA. Die Anpassung an die bei TRANSDATA verwendeten NEA-Protokolle (NEA Netzwerkarchitektur) erfolgt mit Hilfe eines Protokollumsetzers.

Kopplungen zu Datex P

Der Anwender hat die Auswahl zwischen zwei Kommunikationsschnittstellen. Eine Anwenderschnittstelle ist die SINEC-Kommunikationsschnittstelle, die für den Datenaustausch zwischen Anwenderprogrammen innerhalb von SINEC-Rechnernetzen benutzt wird. Der Datenaustausch über das Datex-P-Netz wird zusätzlich durch das SINEC-End-zu-End-Protokoll gesichert.

Die zweite Anwenderschnittstelle setzt direkt auf der genormten X.25/3-Schnittstelle des Datex-P-Netzes auf.

SINEC-*Funktionsverbunde und -Datenverbunde*

Aufbauend auf dem homogenen Nachrichtenverbund, können mit SINEC 300 auch Funktions- bzw. Datenverbunde realisiert werden.

Mit dem SINEC-Funktions- und -Datenverbund wird die bestehende Anwender-Betriebssystemschnittstelle beibehalten, dem Anwender werden aber trotzdem entfernte Eingabe-Ausgabe-(EA-)Geräte und Datenbestände des Rechnerverbunds wie lokale Betriebsmittel zugänglich gemacht (Bild 2). Bestehende Anwenderprogramme und ein breites Spektrum an System- und Dienstprogrammen können damit unverändert im SINEC-Funktionsverbund eingesetzt werden.

Systemstruktur und Funktionen

SINEC ist ein modulares Kommunikationssystem mit mehreren aufeinander aufbauenden Funktionsschichten (Bild 3). Jede Funktionsschicht übernimmt Teilaufgaben des Datenaustausches zwischen den gekoppelten Rechnern.

Die Koordinierung zwischen den Schichten innerhalb eines Rechners erfolgt über einheitliche Schnittstellen, die Koordinierung gleichrangiger Schichten in verschiedenen Rechnern über Protokolle.

Datenübertragungssteuerung

Zur physikalischen Kopplung der einzelnen Rechner stehen mehrere Datenübertragungssteuerungen (DUST) zur Verfügung, die entsprechend den verschiedenen Kopplungsfällen (homogener oder heterogener Verbund, Nah-

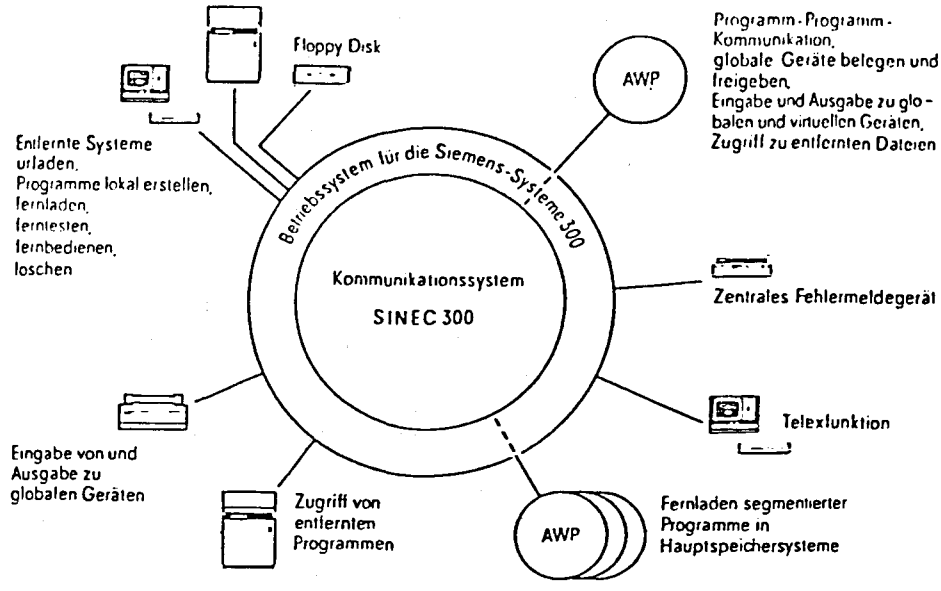


Bild 2
SINEC-Funktions- und -Datenverbund

AWP Anwenderprogramm

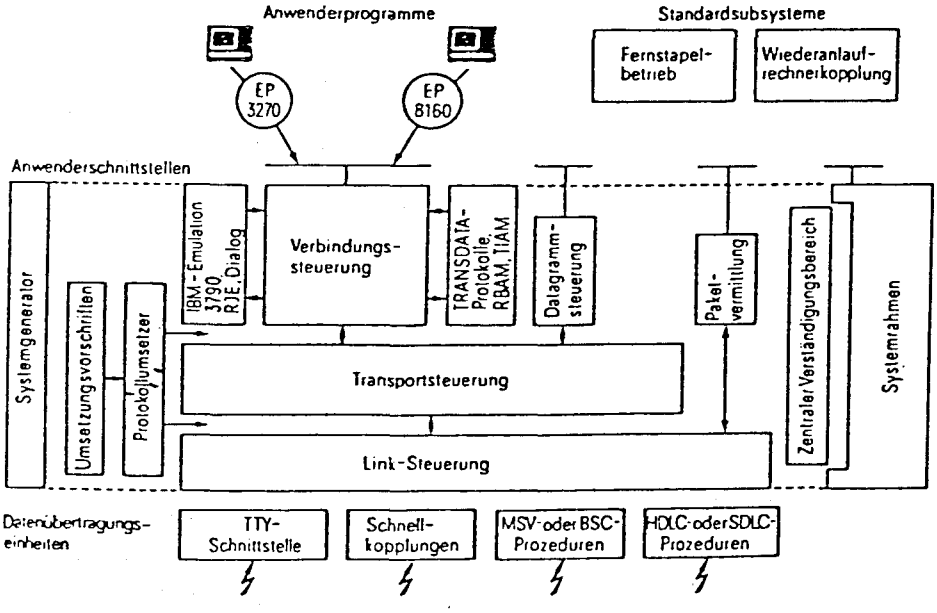


Bild 3
Übersicht über ein modulares Kommunikationssystem mit mehreren aufeinander aufbauenden Funktionsschichten

EP Emulationsprogramm
Weitere Erläuterungen s. Text

oder Fernkopplung) ohne Rückwirkungen auf die Subsysteme eingesetzt werden können. Beim homogenen Verbund können im Nahbereich Datenübertragungsraten bis zu $340 \cdot 2^{10}$ byte/s erreicht werden.

Für die Fernkopplung von Systemen 300 untereinander und für die Kopplung zu Fremdsystemen stehen Datenübertragungssteuerungen mit genormten Leitungsprotokollen (z.B. MSV 1, MSV 2 und BSC sowie HDLC und SDLC) zur Verfügung.

Link-Steuerung

Die SINEC-Link-Steuerung (SNLS) verwaltet und betreibt die am Rechner angeschlossenen Leitungen und wickelt den Datenaustausch zwischen benachbarten Rechnern ab. Bei der Kopplung zu Fremdsystemen, die nicht die SINEC-Protokolle benutzen, erfolgt innerhalb der Link-Steuerung eine Protokollumsetzung.

Transportsteuerung

Die Transportsteuerung (SNTS) übernimmt den Nachrichtentransport zwischen den Subsystemen in den gekoppelten Rechnern mit Hilfe des Transportprotokolls. Dabei werden folgende Teilaufgaben erfüllt:

Multiplexen und Demultiplexen

Die Transportsteuerung verteilt im Quellrechner die Sendewünsche der Subsysteme auf die Ausgabeleitungen und ordnet ankommende Nachrichten den Empfängern zu.

Routing mit Ersatzleitungsumschaltung

Die Zuordnung der Nachrichten zu den Ausgabeleitungen erfolgt über Routingtabellen. Falls die bevorzugte Ausgabeleitung gestört ist, werden die Nachrichten über eine Ersatzleitung weitervermittelt.

Quittierung

Störungen beim Nachrichtentransport werden dem Absender mit negativen Transportquittungen angezeigt.

Verbindungssteuerung

Die SINEC-Verbindungssteuerung (SNVS) stellt das Bindeglied zwischen der Transportsteuerung und den Subsystemen dar und bietet dem Anwender eine leistungsfähige und universelle Kommunikationsschnittstelle an. SNVS übernimmt die Kontrolle über die End-zu-End-Datenübertragung zwischen den Subsystemen und enthält, aufbauend auf die Transportsteuerung, eine Reihe von Zusatzfunktionen, die Voraussetzung für einen störungsfreien Datenaustausch in einem Rechnernetz sind.

SNVS benutzt dazu ein End-zu-End-Transportprotokoll. Die Kommunikation zwischen den Subsystemen geschieht über logische Verbindungen, die durch SNVS verwaltet werden. Die Adressierung der Verbindungspartner erfolgt bei Verbindungsaufbau über symbolische Adressen (Namen).

Durch diese symbolische Adressierung bleiben die Subsysteme unabhängig von der Netzkonfiguration und der Verteilung der Subsysteme innerhalb des Rechnerverbunds.

SNVS unterstützt durch eine Reihe von Maßnahmen den gesicherten Datenaustausch:

Flußsteuerung

Die Flußsteuerung verhindert die Überlastung der Nachrichtenwege dadurch, daß der Sender nur eine bestimmte Anzahl nicht quittierter Nachrichten abschicken darf, bevor er von SNVS so lange angehalten wird, bis die Quittungen eingetroffen sind (Fenstertechnik).

Sequenzkontrolle

Bedingt durch das Routingverfahren, kann es vorkommen, daß sich Nachrichten innerhalb des Netzes überholen. Durch Nachrichtennumerierung und Kontrolle der Laufnummern ist gewährleistet, daß der Empfänger die Nachrichten in der gleichen Reihenfolge erhält, wie sie der Sender abgeschickt hat.

Segmentierung und Reassemblierung

SNVS überträgt Nachrichten, die eine bestimmte Länge (Generierparameter) überschreiten, in mehreren Teilnachrichten zum Zielrechner, setzt die Nachricht dort wieder zusammen und übergibt sie danach an das Empfangssubsystem.

Verbindungsüberwachung und Synchronisierung

Alle Transportvorgänge zwischen den Verbindungspartnern werden zeitüberwacht. Falls eine Überwachungszeit abläuft (time-out), versucht SNVS, die Ursache der Störung durch Synchronisierungsmaßnahmen (z.B. Blockwiederholung) zu beheben.

Datagrammsteuerung

Insbesondere für einfache Punkt-zu-Punkt-Kopplungen steht dem Anwender zusätzlich zu SNVS mit der Datagrammsteuerung (SNDS) eine aufwandarme Schnittstelle zur Verfügung.

Paketvermittlungssteuerung

Der Zugang zu öffentlichen Paketvermittlungsnetzen, gemäß den X.25-Empfehlungen, kann mit SINEC auf zwei verschiedene Arten erfolgen:

Virtuelle Leitung

Die Komponente SNVL realisiert die Funktionen von X.25, Ebene 3, und benutzt dabei permanente (PVC) oder dynamische logische X.25-Verbindungen (SVC) anstelle von physikalischen Leitungen zwischen den Rechnern. Oberhalb der X.25-Protokolle werden die SINEC-Transport- und -Verbindungsprotokolle verwendet.

Paketvermittlung

Mit der Komponente SNPV wird dem Anwender die Ebene 3 von X.25 direkt zugänglich gemacht, wobei er die höheren Schichten und deren Protokolle selbst definieren kann.

Systemrahmen

Die verschiedenen Komponenten von SINEC benötigen eine Reihe gemeinsamer Funktionen, die zu einem Systemrahmen zusammengefaßt sind. Diese Funktionen können auch vom Anwender direkt angesprochen werden. Die wichtigsten Zentralfunktionen sind:

- Puffersystem zur Verwaltung und dynamischen Vergabe von Hauptspeicher- und Externspeicherpuffern,
- Zeitüberwachung zur zeitlichen Kontrolle des Datenaustausches und Überwachung der logischen Verbindungen,
- Fehlermeldung zur Protokollierung von Systemstörungen,
- Systemgenerator,
- Testsystem für Inbetriebnahme und Diagnose sowie
- Administration für Systemauskünfte und -bedienungen, insbesondere zum Nachführen von Änderungen oder Erweiterungen der Netzkonfigurationen.

Schlußbetrachtung

SINEC 300 basiert auf einem dezentralen Rechnernetzkonzept.

Die Topologie eines Rechnernetzes kann somit vom Anwender entsprechend seinen Erfordernissen uneingeschränkt gewählt werden. Problemlos lassen sich auch bestehende Rechnernetzkonfigurationen erweitern oder verändern.

SINEC 300 ist ein mehrlagiges Kommunikationssystem. Der schichtenweise Aufbau der Software (Verbindungssteuerungs-, Transportsteuerungs-, Link-Steuerungs-Ebene) mit einheitlichen Ebenenschnittstellen gewährleistet, daß SINEC 300 zukünftigen Technologien, Standards und Hardware-Produkten angepaßt werden kann, ohne die vom Anwender genutzte Schnittstelle zu beeinträchtigen.

SIGRIS - Siemens Graphisches Interaktives System

System für graphische Datenverarbeitung mit dem Prozeßrechner der R-Serie

Gerhard Just
Siemens AG, Karlsruhe, Abt. E 615

Inhalt:

SIGRIS - System für graphische Datenverarbeitung mit dem Prozeßrechnersystem 300-16

1. Einleitung

Siemens ist, wie Sie sicher wissen, selbst ein potenter Anwender der graphischen Datenverarbeitung und ist für die Hersteller graphischer Geräte und Systeme ein großer und interessanter Markt.
In den letzten sieben Jahren wurden bei Siemens ca. 60 graphische Arbeitsplätze installiert, davon waren allein 50 Systeme von Fremdherstellern wie CV, REDAC, Applicon, Gerber, CDC usw. angekauft bzw. angemietet worden.

Eingesetzt wurden diese Systeme für

den Entwurf integrierter Schaltungen	ca. 15 Systeme
das Entflechten von Leiterplatten	ca. 18 Systeme
das Erstellen von Symbolplänen (Schaltplan, Installationsplan usw.)	ca. 15 Systeme
die mechanische Konstruktion	ca. 5 Systeme
die NC-Programmierung	ca. 5 Systeme
die Automatisierung im Versuchsfeld	ca. 3 Systeme
die graphische Programmiersprachenentwicklung	1 System

Siemens ist und muß daran interessiert sein, daß Entwicklungsmethoden und -ergebnisse nicht der Konkurrenz bekannt werden. Aus diesem Grund hat Siemens auch schon vor Jahren damit begonnen, graphische Datenverarbeitungssysteme für den eigenen Bedarf zu entwickeln. Einige Systeme sind auch auf dem EDV-Markt vertrieben worden bzw. werden noch vertrieben.

Diese seien hier kurz genannt:

a) Graphische Methoden Bank	GMB
b) Interaktives graphisches System	IGS
c) Siemens Graphisches interaktives System	SIGRIS

Auf das System GMB möchte ich nicht weiter eingehen, da auf der SAK-Tagung 1979, Herr Weiß von der TU Wien, das System bereits vorgestellt hat.

Auf das System IGS möchte ich eingehen, da das System SIGRIS hier als Terminal zum Einsatz kommt.
Entstanden ist das IGS durch die Entwicklung einer Plottersoftware für Mikrofilmplotter und mech. Zeichenmaschinen. Aufbauend auf diese Grundsoftware wurden Anwenderprogramme und die Sprache "PICTRAN" zur symbolischen Beschreibung von Zeichnungen einschl. einer Bildbank (Symbolbank) entwickelt.

Es zeigte sich, daß mit diesen Programmen noch keine Zeichnungen mit vertretbarem Aufwand erfaßt und erstellt werden können. So wurde zwangsläufig das IGS entwickelt, welches den Aufbau und die Korrektur von graphischen Daten im Dialog am Bildschirm ermöglicht.

Im IGS sind ca. 100 Kommandos realisiert mit denen:

im Dialog eine graphische Darstellung

- aufgebaut
- geändert und variiert
- angezeigt
- abgespeichert werden kann

im Dialog graphische Elemente

- erzeugt
- kopiert
- verschoben, verdreht, vergrößert, verkleinert und gelöscht werden können

im Dialog Zeichnungen

- aufgerufen und gespeichert
- auf Plotter ausgegeben
- in Ausschnitten dargestellt
- über Menütechnik erstellt werden können.

Folgende graphische Elemente sind standardmäßig realisiert:

- Linien
- Kreise, Kreisbögen
- Rechtecke
- Bildbankbild
- Texte (Textgenerator).

Das IGS ist ein geschlossenes, abläuffähiges Programmpaket und läuft auf der DVA 4004/7000 im BS 2000.

Der graphische Arbeitsplatz des IGS, bis zur Hannover Messe 1980 ein Terminalrechner PR 310K, mit den Aufgaben:

o Speicherung einer Bildinformation mit der Möglichkeit über Änderungsmitteilungen durch DVA 4004/7000 das gespeicherte Bild zu modifizieren.

o Funktion der Textgeneratoren für Softwareschrift

o Funktion eines Kreisinterpolators

o Steuerung des Datenaustausches zwischen Bildschirm, Tastatur, Tablett und DVA (Host)

wird nun durch das System SIGRIS in seiner Grundversion ersetzt.

2. Übersicht

Graphische Systeme bauen zwangsläufig auf Arbeitsplätze auf, die mit großer Interaktivität ausgerüstet sind.

Was gehört zu einem leistungsfähigen graphischen System:

a) ein leistungsfähiger Rechner bzw. ein Verbundsystem

b) eine leistungsfähige graphische Peripherie

c) eine modulare Software, welche eine Interaktivität mit kurzen Antwortzeiten garantiert.

Leistungsfähige Rechner kann Siemens sicher aus der Modellreihe PR 300/R10 - R40 anbieten. Als Design kann sowohl die Schrankversion oder vorzugsweise die SICOMP-Version zum Einsatz kommen. Eine leistungsfähige graphische Peripherie stellt das PR 300 System nicht als Standardgerät zur Verfügung. Es wurde deshalb eine Auswahl der auf dem Markt befindlichen Geräte angeschlossen bzw. deren Anschluß vorbereitet.

Es sind dieses die Gerätetypen:

a) Tablett bzw. Digitizer

b) Bildschirmgeräte

c) Ausgabeeinheiten (Plotter, Hardcopy).

Eine modulare Software stellt das Paket SIGRIS dar, deren Grundversion z. Z. erfolgreich beim IGS zum Einsatz kommt.

3. Systemaufbau-SIGRIS

Das System wird aus bewährten Standardkomponenten zusammengestellt (Bild).

3.1 Besondere Merkmale

- Das Design des Graphik-Arbeitsplatzes ist auf ergonomische Gesichtspunkte ausgerichtet. Die Einzelkomponenten (Tastatur, Tablett, Bildschirm) werden auf Wunsch fest eingebaut oder beweglich angeordnet. Sie können so der jeweiligen Arbeitssituation angepaßt werden.

- Die Bildschirmeinheit zeichnet sich durch hohes Auflösungsvermögen, hohe Darstellungskapazität und absolut flimmerfreie Information aus. Die darzustellende Grafik oder ein Text kann sowohl im Store - wie im Refresh-Modus auf dem Bildschirm angezeigt werden.

- Die Eingabeelemente (Tastatur und Graphik-Tablett) erlauben eine problemorientierte, einfache und besonders effektive Arbeitsweise (Menü-Technik).

- o Die Ausgabe der graphischen Information auf Papier kann durch eine anschließbare Hardcopyeinheit unmittelbar am Arbeitsplatz erfolgen.

- o Plotteranschluß für mehrere Typen ist vorhanden oder kann ohne großen Aufwand realisiert werden.

3.2 Aufbau

Der Graphik-Arbeitsplatz besteht in der Standardkonfiguration aus:

- o Arbeitsplatzrechner SICOMP R10-R30

- o Graphik-Bildschirm

- o Gräphik-Tablett

- o Alpha-num. Bildschirmeinheit mit Tastatur

- o Wahlweise kann eine Hardcopy-Einheit und eine zusätzliche alphanumerische Datensichtstation mit Tastatur oder nur eine zusätzliche Tastatur angeschlossen werden.

- o Die Rechnerkopplung zur nachgeschalteten EDV wird mit der EDUST 3965 realisiert.

3.2.1 Arbeitsplatzrechner

Der Arbeitsplatz enthält einen leistungsfähigen Rechner R10-R30 (R40), aus dem bewährten Siemens-System 300-16 Bit, verbunden mit Plattenspeicher, Magnetband sowie 1 bis 4 Floppy-Disk-Laufwerken und Anschlußeinheiten für Peripherie und Datenkopplung:

. SEAP/GMA-Interface für	Graphik-Bildschirm
. RKE 3964/SIEKON-Interface für	Graphisches Tablett
. EDUST 3965/ (MODEM) für	Datenübertragung

3.2.2 GMA 102 A - Graphische Bildschirmeinheit für SIGRIS-Arbeitsplatz

Als graphisches Display für den SIGRIS-Arbeitsplatz wird das OEM-Produkt GMA 102-A (19"-Diagonale) der Fa. Tektronix - wahlweise GMA 125 (25"-Diagonale) - angeschlossen; diese Bildröhre arbeitet in einem kombinierten Bilddarstellungsverfahren aus Speicher- und Refresh-Technik (STORE-MODE, REFRESH-MODE).

Im STORE-MODE wird das Bild mit erhöhter Strahlenenergie in die Phosphorschicht des Schirms "eingebrannt", d.h., die Schicht wird zum Nachleuchten gebracht.

Im REFRESH-MODE wird der Bildinhalt aus einem Speicher 30 mal in der Sekunde oder mehr, mit energetisch geschwächtem Strahl dem Schirm angeboten. Das Bild (die verschlüsselten Vektoren) ist im Arbeitsspeicher der DVA, dem sogenannten Refresh-Buffer, gespeichert. Er ist max. 8 kWorte lang (sinnvolle Länge für Refresh-Betrieb). Damit kann ein Bild mit einer Vektoranzahl $\lesssim$ 4 k im Refresh-Betrieb dargestellt werden.

Neben einem Software-Zeichengenerator, der als variabler Modul in der SIGRIS-Grundsoftware eingebaut ist, besitzt der Bildschirm einen Hardware-Zeichengenerator, der bei entsprechender Codierung 4 Schriftgrößen in 2 Schriftstärken jeweils senkrecht oder schräg erzeugt.

Für die unmittelbare Ausgabe des im STORE-MODE gespeicherten Bildinhalts auf Kopien kann ein Hardcopygerät angeschlossen werden.

3.2.3 Graphisches Tablett

Die Eingabe der graphischen Daten, der Menü-Funktionen und die Koordinaten des Cursors und des Fadenkreuzes für den Bildschirm wird durch ein Tablett vorgenommen. Die Auflösung dieses Tabletts beträgt 0,1 mm, der Anschluß eines hochauflösenden Digitizers ist vorgesehen.

3.2.3.1 Tablettgrößen

Es stehen 13 verschiedene Tablettgrößen zur Verfügung. Sie reichen von 28 x 28 cm bis zu 106 x 152 cm nutzbarer Fläche. Standardmäßig wird ein Tablett mit einer Meßfläche von 42 x 60 cm eingesetzt.

3.3 Software

Die SIGRIS-Basis-Software umfaßt die Funktionen eines graphischen Betriebssystems:

- Datentransfer mit der graphischen Peripherie
- Verarbeitung graphischer Befehle
- Verwaltung graphischer Elemente im Refresh-Buffer
- Zeichengeneratoren für die Ausgabe von Text
- Datenübertragung und Verbindung zu übergeordneten graphischen DV-Verfahren

Es sind eine Assembler (ASS 300) und eine FORTRAN-Version (FORTRAN 300) implementiert, die auf dem Rechner R10, R20..., ablauffähig sind. Dem Anwender liegt eine Schnittstelle vor, die durch Unterprogrammaufrufe angesprochen wird (z.B. R7: US LINE in ASS 300 bzw. CALL LINE in FORTRAN 300). Ein GKS (Graphisches Kern System nach DIN und ISO) als Erweiterung der SIGRIS-Grundsoftware ist in der Entwicklung. Diese Schnittstelle wird die graphische Anwendersoftware, ausgeführt nach der DIN-Norm, portabel für entsprechende Hardware machen.

3.3.1 Datentransfer

Für den Datentransfer mit der Peripherie sind Treiberbausteine implementiert worden, die auf die logischen und physikalischen Verhaltensweisen der Geräte eingehen. Die Geräte werden durch Aufrufe angesprochen.

3.3.2 Graphische Befehlsverarbeitung

Der Anwender kann durch Eingabe von Parametern und Befehlen graphische Funktionsbausteine (ca. 15) anstoßen z.B. Linien-, Bogen-, Rastergenerator usw. (Befehlsliste siehe Pkt. 3.3.6) und Bausteine für Transformationen, Bildbegrenzung und Anzeigenbildung im Fehlerfall.

3.3.3 Verwaltung graphischer Dateien

Im Rahmen der graphischen Elementeverwaltung werden dem Anwender Aufrufe zur Verfügung gestellt, die dazu dienen:

- Graphische Elemente zu definieren
- Graphische Elemente in den Refreshzyklus einzureihen bzw. herauszunehmen
- Graphische Elemente im STORE-MODE auszugeben usw.

3.3.4 Zeichengenerator

Neben dem Hardwarezeichengenerator der GMA-Röhre, der nur eine Schriftart bietet, die in 4 Größen und 2 Schriftstärken variiert werden kann, ist ein Softwarezeichengenerator eingerichtet. Es können beliebig viele Schriftarten erzeugt werden. Die Zeichen einer Schriftart (z. Zt. ist die Schriftart AV eingerichtet) werden im Lettertable hinterlegt. Größe, Schriftwinkel und Textwinkel werden durch Parametrierung vom Anwender freiprogrammierbar angegeben.

3.3.5 Datenübertragung

Der Software-Baustein für die Kopplung an übergeordnete graphische DV-Verfahren wird durch Aufrufe für z.B. Senden und Empfangen vom Anwender angestoßen. Er kann von mehreren Anwendern parallel betrieben werden.

4. Zusammenfassung

- o Siemens bietet dem Anwender mit dem SIGRIS-System eine Hardware und eine Softwareschnittstelle an, auf deren Basis sich leicht graphische Anwendungen aufbauen lassen. (ASS 300, FORTRAN, GKS-Schnittstelle).
- o SIGRIS bietet die Möglichkeit, sich als I/O-Terminal an graphische DV-Verfahren lauffähig auf einer DVA, anzuschließen (z.B. IGS, REGENT usw.)
- o SIGRIS kommt als SICOMP-Version - Graphischer Arbeitsplatz - zum Einsatz.
- o SIGRIS-Komponenten, wie Bildschirm, Digitizer oder Plottermodule können auch mit den entsprechenden Softwaremodulen an schon installierten PR-Anlagen nachgerüstet werden.
- o Anwendersoftware ist in der Entwicklung.
  Realisiert werden z.B. ein Digitalisier- und Editorsystem, eine NC-Programmiereinheit und ein geschlossenes ablauffähiges interaktives graphisches Programmpaket, ähnlich dem IGS, für das PR-System 300.
- o Die Verwendung der SIGRIS-Grundversion als Terminal für die DV-Verfahren des 7000-Systems, bekannt durch die Namen und Anwendungen
  - . CADIS 2D und 3D für den Konstrukteur
  - . IGS für die Vermessungsaufgaben
  - . IGS für die Netzberechnung und -optimierung im EVU Bereich

  sollen hier nicht unerwähnt bleiben.

Aufrufe-Übersicht

Geräte-Aufrufe

EINIT	- Eingabegerät initiieren
AINIT	- Ausgabegerät (graphisches) umschalten
RESET	- Peripheriegerät rücksetzen
TABKAL	- Tablett kalibrieren
COPY	- Hardcopy von Bildschirm (GMA) abziehen
ERASE	- Bildschirm löschen
BELLS	- Akustisches Signal am Schirm ausgeben
BELLT	- Akustisches Signal am Tablett ausgeben

Ein/-Ausgabe Aufrufe

LESE - Daten von Tablett, Tastatur, Steuerknüppel in einen Ringpuffer eintragen

DSTEXT - Dialogtext am Schirm ausgeben

Graphische Aufrufe

LINE, POSIT, PUNKT - Linie bzw. Punkt am Schirm zeichnen oder Positionen dunkel einstellen

BOGEN - Kreis/Bogen zeichnen

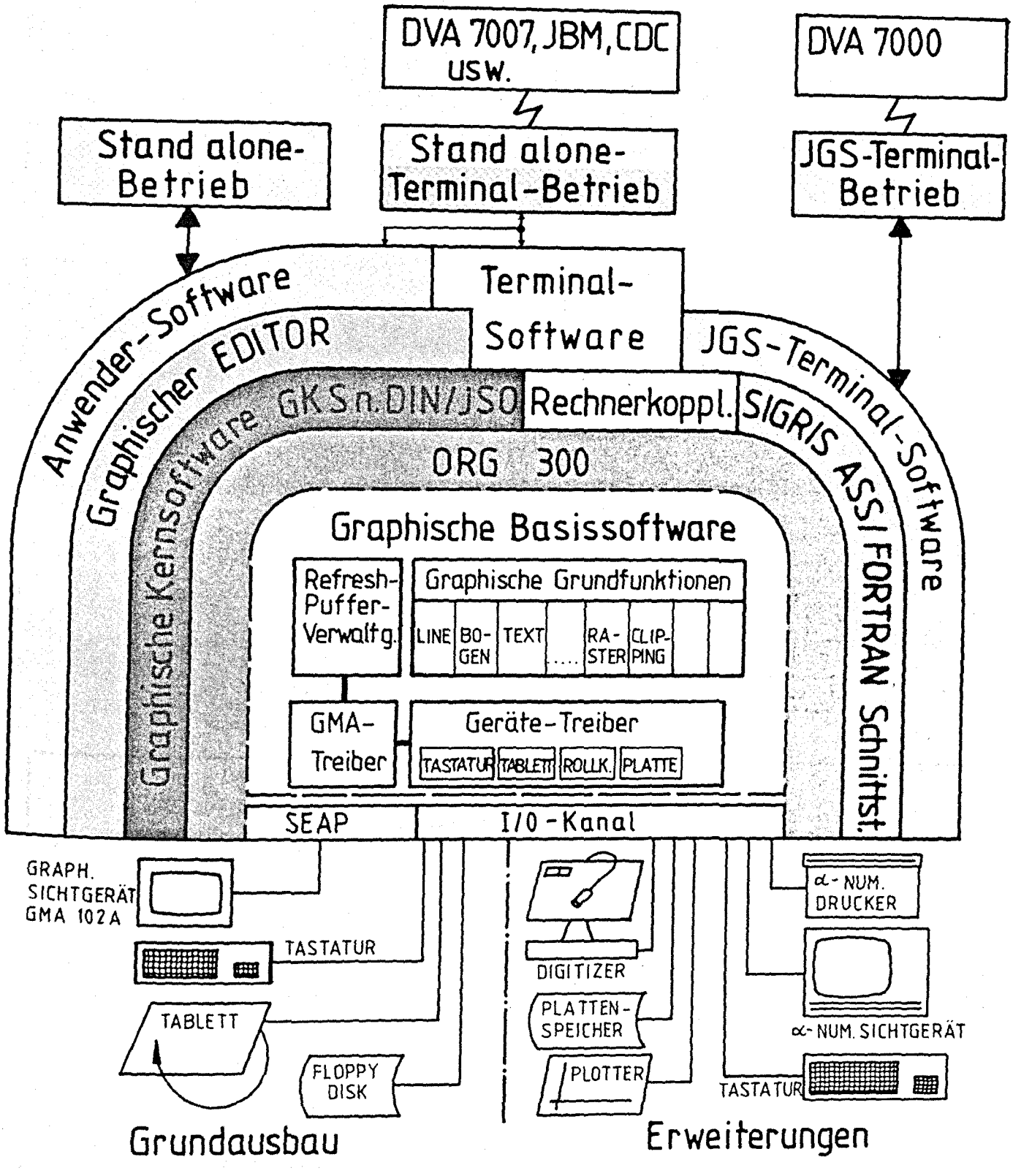
TEXT - Hardware- bzw. Graphiktext ausgeben

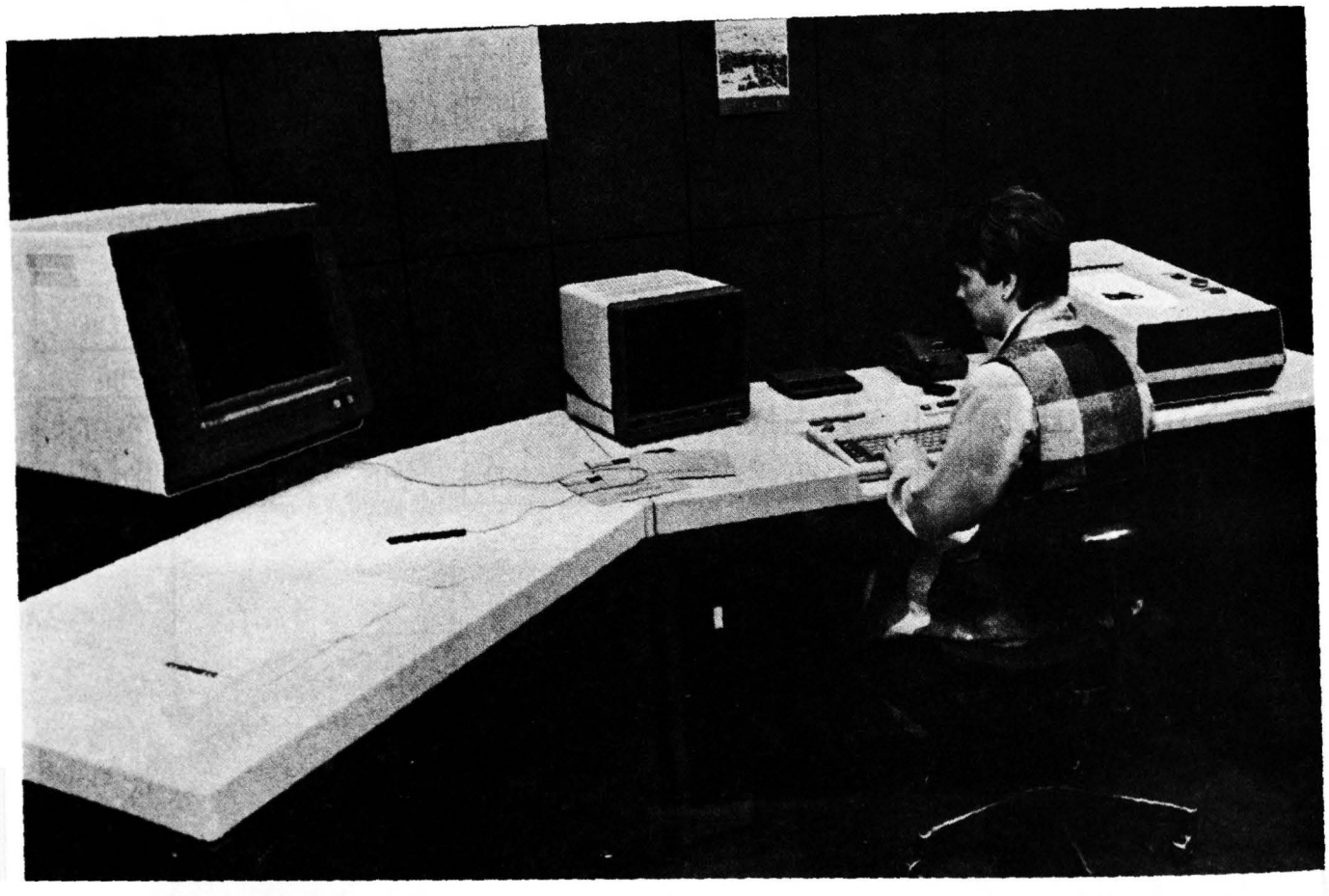
PRAST, MRAST, ARAST - Punkt- bzw. Achsenraster ausgeben

FADE - Fadenkreuz aufrufen

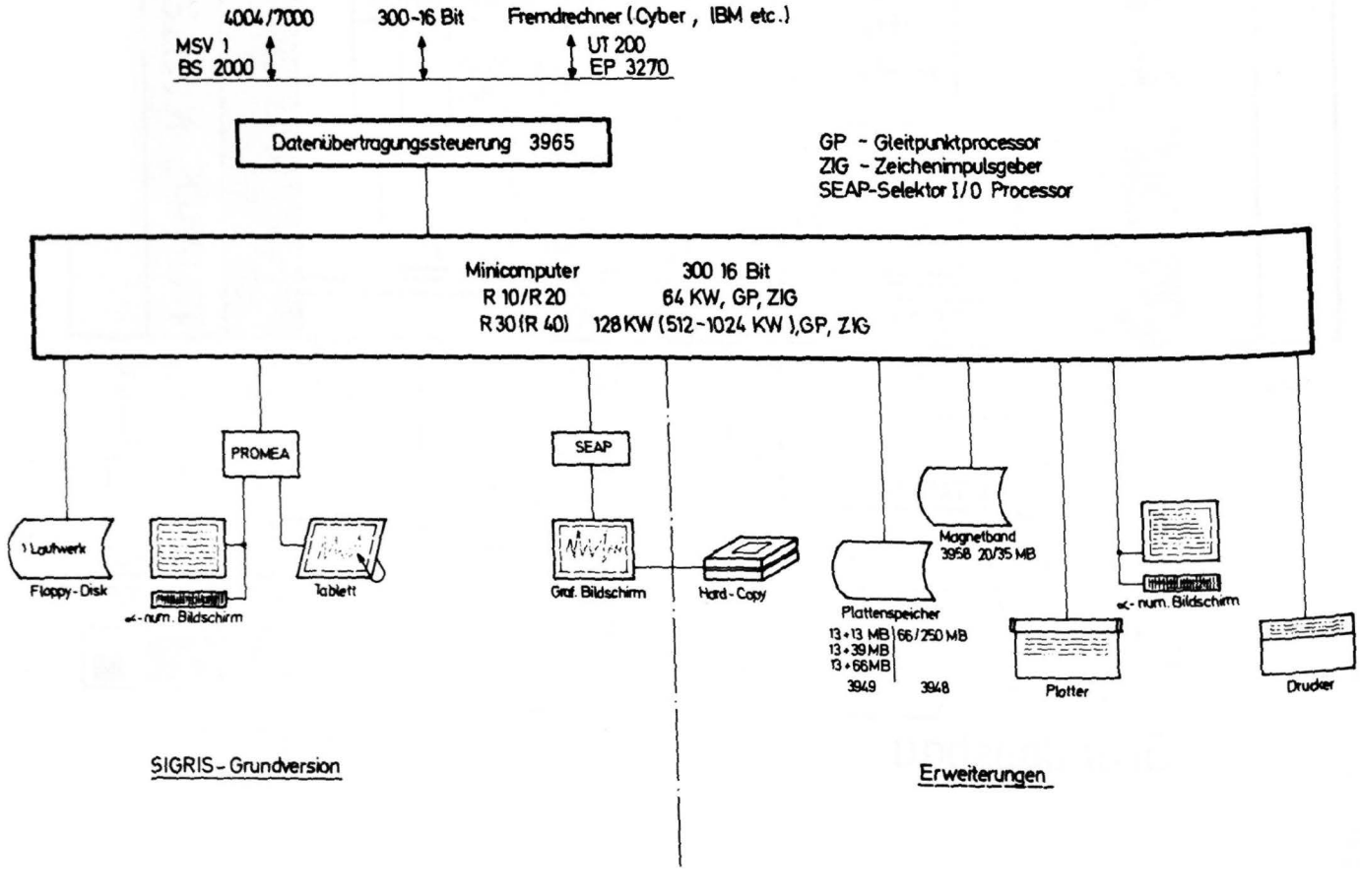
FAPO - Fadenkreuzposition abfragen

Graphisches Datenverarbeitungssystem SIGRIS für CAD/CAM, Labor, Versuchsfeld, Prüfautomatisierung, Planungsaufgaben





Hardware - Konfiguration 'SIGRIS'



PASCAL 300

Eine neue Programmiersprache für die Siemens Systeme 300

Dr. Jürgen Vetter
Siemens AG
Bereich Verarbeitende Industrie
Prozeßrechner Fachabteilung

1. Einleitung

Der Nachweis, daß höhere Programmiersprachen für den Anwender von großem Nutzen sind, muß heute nicht mehr geführt werden. Der Vorteil der Verringerung der Softwarekosten ist offensichtlich. Dies wird in vielfältiger Weise erreicht. Die Verkürzung der Entwicklungszeiten wird durch den konsequenten Gebrauch von höheren Programmiersprachen ebenso möglich, wie die Beschleunigung des Reifeprozesses eines Programmes und damit die Erhöhung der Software-Sicherheit.

Wesentliches tragen höhere Programmiersprachen zu einer guten Portabilität der Anwenderprogramme und ihrer leichten Lesbarkeit bei. Der letzte Gesichtpunkt ist insbesondere im Hinblick auf das stetig größer werdende Problem der Programmdokumentation von Bedeutung. Weiterhin unterstützen einige, in den letzten Jahren zu zunehmender Bedeutung gelangte Programmiersprachen die modernen Entwurfsmethoden des Software-Engineerings. Zu diesen Sprachen gehört auch PASCAL.

2. PASCAL

2.1 Entwurfsziele von PASCAL

Um zu einer Einschätzung einer höheren Programmiersprache zu gelangen, ist es hilfreich zu wissen, was die Erfinder der Sprache selbst erreichen wollten, mit anderen Worten, welches die Entwurfsziele zu dieser Sprache waren. Diese Ziele sind für PASCAL präzise definiert: PASCAL sollte als Lehrsprache zur Programmierausbildung nach modernen Gesichtpunkten dienen

und die leichte Entwicklung effizienter Compiler ermöglichen.

Erfüllt eine Programmiersprache die Anforderung Lehrsprache zu sein, so ist dies äquivalent zu einer Reihe noch zu nennender Eigenschaften, die im besonderen Maße Einfluß auf die Verkürzung von Software-Entwicklungszeiten haben, desgleichen trägt die leichte Portierbarkeit von Compilern zur starken Ausbreitung einer Programmiersprache und, einen gewissen Sprachstandard vorausgesetzt, zu einem hohen Portabilitätsgrad der Anwendersoftware bei.

Welche Merkmale zeichnen nun PASCAL als moderne Lehrsprache aus? Das globale Konzept der blockorientierten Sprache ermöglicht die Programmierung einer Problemstellung ausgehend von der Gesamtaufgabe hin zur Auffächerung auf Teilfunktionen, d.h. den Programmdesign nach dem Top-down-Prinzip. Die Beschränkung auf einige wenige ausführbare Anweisungen führt zu einer übersichtlichen leicht verständlichen Sprache. Eine freie Beschreibbarkeit der Datentypen erlaubt dem Anwender problemadäquate Definitionen. Die Sprachkonstrukte führen zu höherer Prüfbarkeit zur Compile-Time, was die Ausführungszeiten von Objektprogrammen entscheidend verringern hilft.

2.2 Einsatzbereiche von PASCAL

Aus diesen globalen Merkmalen lassen sich die Einsatzbereiche der Sprache bestimmen. PASCAL ist als "general purpose language" nicht auf einen bestimmten Anwendungsbereich festgelegt, sondern bietet sich stets zum Einsatz an, wenn es gilt, nach modernen

Software-Engineerung-Methoden Programme zu erstellen, für die Gesichtpunkte wie Software-Sicherheit, Änderungsfreundlichkeit und Portabilität im Vordergrund stehen. Daher wird PASCAL in immer stärkerem Maße als Systemimplementierungssprache, d.h. zur Erstellung von Organisationsprogrammen, Compilern und systemnahen Dienstprogrammen herangezogen. Hierbei dürfen selbstverständlich Einschränkungen, die durch das Sprachkonzept bedingt sind, wie etwa hinsichtlich der Leistungsfähigkeit der Ein/Ausgabefunktionen nicht übersehen werden.

2.3 Sprachstrukturen von PASCAL

Ein PASCAL-Programm setzt sich aus Programmkopf, Deklarationsteil und Anweisungsteil zusammen. Im Programmkopf sind der Name des Programms und die Parameter des Programms spezifiziert. Diese Parameter bezeichnen Dateien, die in ihrer Existenz vom Programm unabhängig sind und über die Daten mit der Programmumgebung ausgetauscht werden. Im Deklarationsteil werden die Vereinbarungen zu dem Programm getroffen. Es werden die im Programm verwendeten Marken, Konstanten, Typen, Variablen und Prozeduren bzw. Funktionen festgelegt. Der Anweisungsteil enthält die ausführbaren Anweisungen des Programmblocks, die durch die Schlüsselworte BEGIN und END geklammert sind. Jede im Anweisungsteil des Programms eingeführte Prozedur oder Funktion beinhaltet ihrerseits wieder einen Deklartionsteil und einen Anweisungsteil. Dies führt zu einer Programmstruktur aus geschachtelten Blocks. Vereinbarungen haben nur im zugehörigen und diesen untergeordneten Blöcken Gültigkeit.

Ein weiteres wesentliches Charakteristikum der Sprache PASCAL ist die dem Anwender freigestellt Definition eigener problemadäquater Datentypen als sogenannte Aufzählungstypen. Hierbei wird der Wertebereich eines Typs durch die Nennung der Werte definiert wie z.B. Typ Jahreszeiten = (Frühling, Sommer, Herbst, Winter). Zu diesen Typen lassen sich Unterbereichstypen festlegen. Weiterhin können aus bereits definierten Typen die sogenannten strukturierten Typen konstruiert werden. Es stehen vier Strukturierungsmethoden durch die Sprache zur Verfügung: Array-Struktur, Record-Struktur, File-Struktur und Set-Struktur. Mit einer Array-Struktur werden typhomogene Komponenten, mit einer Record-Struktur typ-inhomogene Komponenten zusammengefaßt. Die File-Struktur dient zur Beschreibung von Dateiinhalten. Hier werden ebenfalls typ-homogene Komponenten strukturiert, im Gegensatz zur Definition des Arrays ist die Länge eines Files nicht festgelegt. Insbesondere wird der Zugriff auf den File durch sprachimmanente Definition und Handling eines Filezeigers unterstützt. Über die Set-Struktur wird zum Wertebereich eines Aufzählungstyps die Potenzmenge auf Sprachebene zur Verfügung gestellt. Die notwendigen Mengenoperationen stehen in PASCAL zur Verfügung.

Neben einfachen und strukturierten Typen gehört das Konstrukt des Pointer-Typ zum Sprachumfang. Dieser Typ kann zu Aufbau komplexer Datenstrukturen wie Datenbäumen verwendet werden.

Die ausführbaren Anweisungen beinhalten im wesentlichen einige wenige Kontrollstrukturen, mit deren Hilfe ein Programmentwurf z.B. nach der Jackson-Methode effizient umgesetzt werden kann. Insbesondere trägt die Möglichkeit der rekursiven Aufrufbarkeit von Prozeduren sehr zur Laufzeitoptimierung von mathematischen Programmen wie Iterationsverfahren bei.

2.4 Vorteile von PASCAL

- PASCAL besitzt ein breites Anwendungsspektrum, daß die Programmierung von Aufgabenstellungen sowohl aus dem technisch-wissenschaftlichen wie auch kommerziellen Bereich unterstützt.

- PASCAL ist eine leicht verständliche Programmiersprache, die sich in ihren Sprachkonstrukten auf das Notwendige beschränkt. Diese Konstrukte vermeiden Ausnahmeregelungen.

- PASCAL eignet sich für die Methode der Strukturierten Programmierung und ermöglicht damit einen hohen Überprüfbarkeitsgrad der statischen Programme.

- Die Sprachdefinition ist klar, vollständig und kurz.

- PASCAL ist schnell erlernbar.

- In PASCAL sind Deklarationsteil und Anweisungsteil scharf getrennt. Dies unterstützt eine datenorientierte Programmanalyse und Codierung nach Jackson.

- Durch problemadäquate Beschreibung der Daten werden Informationen aus der Aufgabenstellung über diese verwendeten Daten in die Datendefinition im Programm eingebracht.

- PASCAL-Programme sind in hohem Maße selbstdokumentierend. Ihre Lesbarkeit und Verständlichkeit ist sehr gut.

Im Vergleich zu anderen bekannten und gebräuchlichen Programmiersprachen wie FORTRAN, BASIC und COBOL zeigen sich hinsichtlich der einzelnen Produktphasen der Software wichtige Unterschiede. So ist hinsichtlich der Lernbarkeit BASIC PASCAL sicherlich überlegen. in den Punkten Entwurfsrealisierung und Wartbarkeit ist

PASCAL am besten geeignet. Anforderungen an die Portabilität der Anwenderprogramme werden durch FORTRAN und COBOL aufgrund deren hohen Normierungsgrades und weiten Verbreitung am besten erfüllt. In allen vier Phasen zeigt PASCAL jedoch eine überdurchschnittlich gute Eignung.

2.5 Schwachstellen von PASCAL

Zum ursprünglichen Entwurf von PASCAL, dem sogenannten Standard-PASCAL, das im Report von Jensen-Wirth festgelegt ist und das auch heute noch für Neuimplementierungen einen hohen Verbindlichkeitsgrad besitzt, sind natürlich Verbesserungen möglich. So fehlen Sprachelemente, die einen komfortablen direkten Dateizugriff ermöglichen. Ohne das Sprachkonzept sprengen zu müssen, sollten weiterhin implementiert werden der Datentyp String, die Möglichkeit dynamischer Feldgrenzen und die Initialisierung der Variablen bei ihrer Deklaration. Der Begriff der externen Prozedur fehlt ebenso, wie ein Konzept zum Anschluß von Moduln zu einem PASCAL-Programm ebenfalls nicht unterstützt wird.

3. PASCAL 300 für die Siemens Systeme 300

Für die Siemens Systeme 300 wird mit PASCAL 300 eine Obermenge des im Jensen-Wirth Report beschriebenen Sprachumfangs von Standard PASCAL zur Verfügung gestellt. Die über Standard PASCAL hinausgehenden Sprachkonstrukte von PASCAL 300 werden aufgrund folgender Überlegungen eingebracht:

- Einer nationalen oder internationalen Normierung von PASCAL soll nicht vorgegriffen werden, d.h. Sprachkonstrukte, die im Report beschrieben sind und keinen Bezug auf spezifische Eigenschaften des Systems 300 haben, werden nicht verändert oder erweitert.

- Implementierungsentscheidungen werden in Abstimmung der PASCAL-Entwicklungen der Systeme Siemens 7000, Siemens 300 und des Mikroprozessorsystems SME getroffen, um die Portabilität der Programme zu gewährleisten.

- Systemabhängige Konstrukte werden eingebracht, um die beschriebenen Schwachstellen von Standard PASCAL auszumerzen.

Für PASCAL 300 wurden die folgenden Erweiterungen implementiert:

- Serielle Ein/Ausgabe auf ORG-Dateien und Bibliothekselemente
- Direktzugriff auf ORG-Dateien durch Angabe der Satznummer
- Realisierung des Bedien-Aufrufes auf Sprachebene
- Kennzeichnung von Prozeduren auf Sprachebene, die in getrennte Moduln übersetzt werden sollen. Dies ermöglicht die Definition einer Overlay-Struktur zu einem PASCAL 300-Programm.
- Anschlußmöglichkeit von externen Routinen in den Sprachen FORTRAN 300, COBOL 300, BASIC 300 und Assembler ASS 300.

Die genannten Funktionen gewährleisten eine reibungslose Einpassung von PASCAL 300-Programmen in die Umwelt der Systeme 300.

4. Eigenschaften des PASCAL 300-Compilers

Der Compiler ist ablauffähig auf den Minicomputern R10 bis R40. Als reentranter Ausgabecode des Compilers kann der Anwender wählen:

- Ausgabe in noch zu bindender Grundsprache GS 300
- Ausgabe in bereits lade- und ablauffähiger Grundsprache GS 300
- Ausgabe in Assemblersprache ASS 300.

Der Compiler führt lokale Code-Optimierungen durch wie:

- Vorberechnung konstanter Ausdrücke
- Registeroptimierung zur Abarbeitung arithmetischer Ausdrücke
- Optimierung Bool-scher Ausdrücke.

Zur Testerleichterung von PASCAL-Programmen stellt der Compiler folgende Testhilfen zur Verfügung:

- Prozedur-Backtrace, d.h. Rückverfolgung der dynamischen Aufrufvergangenheit durch Ausgabe der Namen durchlaufener Prozeduren
- Überwachung arithmetischer Überläufe
- Dynamische Überwachung von z.B. Subrange-Variablen und Feldgrenzen.

5. Schlußbetrachtung

Mit der Programmiersprache PASCAL 300 und dem zugehörigen Compiler wird dem Anwender der Siemens Systeme 300 ein zusätzliches Mittel zur Hand gegeben, auf der Ebene höherer Programmiersprachen unter Berücksichtigung moderner Software-Engineering Methoden kostengünstig Softwareprodukte zu erstellen.

Leistungsumfang und Einsatzgebiet der zwei FORTRAN-Compiler der Siemens Systeme 300

Dr. Jürgen Vetter
Siemens AG
Bereich Verarbeitende Industrie
Prozeßrechner Fachabteilung

1. Einleitung

Für die Siemens Systeme 300 stellt FORTRAN die am häufigsten verwendete höhere Programmiersprache dar. Dies liegt an der ursprünglichen Ausrichtung der Systeme 300 auf prozeßtechnische und technisch-wissenschaftliche Aufgabenstellungen. So wurde zu der heutigen 16-Bit-Modellreihe schon sehr frühzeitig mit dem FORTRAN-Compiler FC 30 ein Werkzeug zur Erstellung von FORTRAN-Programmen zur Verfügung gestellt. Der stetigen Entwicklung des Software-Engineerings folgend, wird nun mit dem FORTRAN 300-Compiler ein Produkt freigegeben, in dessen Entwicklung die aus der Arbeit mit dem FC 30 gewonnenen Erkenntnisse eingegangen sind und das zum anderen in seinen Eigenschaften die aktuellen Marktdaten berücksichtigt.

2. FORTRAN 300

2.1 Zielsetzung von FORTRAN 300 in Abgrenzung zu der Sprache des FC 30

Parallel zu der Entwicklung der Siemens Systeme 300 von der nahezu alleinigen Anwendung als Prozeßrechner zum vielseitig verwendbaren Minicomputer ist auch in den Anforderungen an den Leistungsumfang der verfügbaren höheren Programmiersprachen ein Wandel eingetreten. So stand bei der Entwicklung des Compilers FC 30 sowohl im Hinblick auf den Compiler als auch im Hinblick auf die durch ihn erzeugten Objektprogramme die Prozeßanwendung im Vordergrund. Daher ist der durch den FC 30 bearbeitete Sprachumfang durch zahlreiche Elemente

zum Verkehr und der Reaktion auf den Prozeß ausgestattet, ist der abgesetzte Code durch umfangreiche Prüfroutinen ausgezeichnet und ist der Compiler selbst mit Laufeigenschaften als Hintergrundprogramm simultan zum Prozeß ausgelegt.

Diese Leistungsdaten treten beim Compiler FORTRAN 300 in den Hintergrund zugunsten einer allgemeinen Verwendbarkeit im technisch-wissenschaftlichen und z.T. auch kommerziellen Bereich. Wesentlich hierbei ist jedoch, daß die Sprache FORTRAN selbst durch die neue Norm von 1978 sehr stark verbessert wurde und ein breiteres Einsatzspektrum gewonnen hat. Die Verfügbarkeit dieser neuen Norm durch FORTRAN 300 ist ein wichtiges Leistungsplus.

2.2 Der Sprachumfang von FORTRAN 300

Um den Sprachumfang von FORTRAN 300 skizzieren zu können, ist es notwendig, einige Möglichkeiten der neuen Norm ANSI - X3.9 - 1978 zu beleuchten.

So waren für den Entwurf der Norm die folgenden Kriterien entscheidend:

- Aufnahme von Sprachelementen, deren Nützlichkeit durch zahlreiche Implementierungen bewiesen war
- Aufnahme von der die Portabilität erhöhende Sprachkonstrukte
- Möglichst geringe Vergrößerung der Komplexität der Sprache

. Vermeidung von Widersprüchen zur alten Norm von 1966

. Verzichte auf Sprachelemente der Norm '66 nur aus schwerwiegenden Gründen

. Präzisere Beschreibung der Sprache

Die Norm ist jedoch permissiv in dem Sinne, daß sie nicht festlegt, was ein Compiler mit Sprachelementen tun soll, die nicht der Norm entsprechen.

Die Norm definiert eine Untermenge, das sogenannte SUBSET-FORTRAN '78, ebenfalls ein wichtiger Beitrag zur Portabilität.

Im folgenden werden beispielsweise einige gegenüber der alten Norm zusätzliche Sprachelemente aus der Norm '78 genannt:

. Ersetzung des Datentyps "Hollerith" durch den Datentyp "Character" und Einführung eines modernen Stringkonzeptes

. Gemischte Arithmetik - Ausdrücke stehen nun an Stellen, wo vorher nur Konstanten zugelassen waren, wie bei Feldgrenzen oder den Parametern des DO-Statements

. Unterstützung der Methoden der strukturierten Programmierung durch das Konstrukt eines Block - IF

. Erweiterung des Ein-/Ausgabekonzeptes durch:
  .. Direktzugriff auf Dateien
  .. Spezifizierung eines Default-Gerätes und -Formates
  .. Eröffnen und Schließen von Dateien
  .. Abfragen des Datei-Status
  .. Handling von Strings analog zu Dateien.

Für die echte Untermenge SUBSET-FORTRAN gilt, daß sie an diesen und allen weiteren wichtigen Sprachkonstrukten von FULL-FORTRAN '78, die den Einsatzbereich der Sprache erweitern, wesentlich Anteil hat.

Für die Festlegung des Sprachumfangs von FORTRAN 300 waren hauptsächlich die beiden folgenden Gedanken maßgebend:

. Verfügbarkeit der aus der Sicht der Systeme 300 und vergleichbarer MDT-Rechner wichtigen Sprachkonstrukte der Norm '78

. Leichte Portierbarkeit von Programmen ohne Prozeßanwendungen aus der Sprache des FC 30 in FORTRAN 300.

Die erste Überlegung führte dazu, eine große Obermenge von SUBSET FORTRAN in Richtung FULL FORTRAN in FORTRAN 300 zu realisieren. So sind über SUBSET FORTRAN hinaus dem Anwender die folgenden Konstrukte aus FULL FORTRAN zur Verfügung gestellt:

. die Datentypen DOUBLE PRECISION und COMPLEX

. die Anweisungen BLOCK DATA, ENTRY, CLOSE, INQUIRE

. die logischen Operatoren .EQV. und .NEQV.

. listengesteuerte Ein-/Ausgabe

. erweiterte OPEN- und READ-Anweisung

. 19 mögliche Fortsetzkarten.

Der zweite Gedanke des leichten Übergangs vom Compiler FC 30 zum FORTRAN 300 - Compiler führte zur Übernahme der folgenden Sprachelemente aus dem Verarbeitungsumfang des FC 30 in FORTRAN 300:

. die Datentypen INTEGER *2, LOGICAL *2

. die Anweisung DEFINE FILE

. Integergrößen als Argumente logischer Operationen

. Steuerzeichen zur Formularsteuerung.

So soll sowohl für Alt- als auch Neuanwender der Gebrauch von FORTRAN mit den Systemen 300 auf hohem Niveau problemlos ermöglicht werden.

Eigenschaften des FORTRAN 300-Compilers

Der Compiler ist ablauffähig auf den Minicomputern R10 bis R40. Als Ausgabecode des Compilers kann der Anwender wählen:

. Ausgabe in noch zu bindender Grundsprache GS 300

. Ausgabe in bereits lade- und ablauffähiger Grundsprache GS 300.

Diese zweite sehr hantierungsfreundliche und schnelle Methode, ein ablauffähiges FORTRAN-Programm zu erstellen, wird durch einen in den Compiler integrierten Linearbinder ermöglicht, der eine interne Zwischensprache des Compilers weiterverarbeitet. Bei Programmen, für die aufgrund ihres Umfangs eine Overlay-Struktur vorteilhafter ist, wird man den erstgenannten Weg der Ausgabe noch zu bindender Grundsprache wählen. Diese wird mit dem vom Compiler unabhängigen Systemprogramm Binder BD 30 in ladbare Form mit gewünschtem Überlagerungsaufbau überführt.

Der Wunsch nach anwendungsfreundlicher Hantierung führte zur Gestaltung des übersichtlichen Übersetzungsprotokolls und der differenzierenden Fehlerdiagnose.

Ein wesentliches Entwurfsziel war neben der bereits erwähnten Benutzerfreundlichkeit der schnelle Übersetzungsvorgang.
Dies wird durch die Konzipierung als 2-Paß-Compiler erreicht. Die Vorberechnung konstanter Ausdrücke durch den Compiler beschleunigt darüber hinaus den Ablauf der Objektprogramme.

Durch das Einfügen sogenannter DEBUG-Zeilen in das Programm hat der Anwender ein Mittel zur Programmablaufverfolgung zur Hand. Diese DEBUG-Zeilen werden auf Anwenderwunsch durch den Compiler in die Grundsprache des Programms überführt.

Einsatzbereiche der Compiler FC 30 und FORTRAN 300

Die Einsatzgebiete der beiden FORTRAN-Compiler und der durch sie zur Verfügung gestellten Sprachvarianten sind hinreichend unterschiedlich, um die Existenz zweier Compiler zu rechtfertigen.

Mit dem Compiler FC 30 erschließt sich dem FORTRAN-Anwender die Programmierung komplexer Prozeßabläufe von der Netzautomatisierung bis zur Walzwerksteuerung. Leistungsfähige Sprachelemente zum Verkehr mit der Prozeßperipherie,gestützt durch komfortable Funktionen des Bithandlings und der Zugriff auf die Taskingfunktionen des Betriebssystems, ergänzt um das problemlose Mischen mit Assemblercode, stellen für die obengenannten Aufgaben mächtige Hilfen dar. Demgegenüber wird der

FORTRAN 300-Compiler von denjenigen Anwendern genutzt werden, die die Sprache FORTRAN in ihrer klassischen Anwendung der technisch-wissenschaftlichen Programmierung auf neuestem Niveau nutzen wollen und kurze Übersetzungszeiten und hohen Hantierungskomfort erwarten.

5. Schlußbetrachtung

Mit der parallelen Verfügbarkeit zweier FORTRAN-Compiler wird dem hohen Stellenwert dieser Sprache für die Systeme 300 entsprochen. Neben der Unterstützung der prozeßtechnischen und der ausschließlich technisch-wissenschaftlichen Anwendung wird auch denjenigen Anwendern Hilfestellung gegeben, die aufgrund ihrer Aufgabenstellung vom FC 30 zum FORTRAN 300-Compiler wechseln wollen. So fügt sich auch dieser neue Compiler in die Palette der Sprachübersetzer der Siemens Systeme 300 nahtlos ein.

Allgemeine Struktur von Programmsystemen für Meßaufgaben; Programmbausteine und Kommando-Interpreter

K. Weise, R. Rybarsch, H.-J. Schuster

Physikalisch-Technische Bundesanstalt, Braunschweig

1. Übersicht

Die Physikalisch-Technische Bundesanstalt (PTB) in Braunschweig besitzt für die Meßprozeß-Automatisierung bei vielen (ca. 100) sehr verschiedenartigen Meß- und Prüfaufgaben ein Prozeßrechner-Verbundsystem, das derzeit aus elf Siemens-Prozeßrechnern 330, weiteren sechs Siemens-Rechnern R30 im Aufbau und etwa 60 Kleinrechnern Tektronix 4051/4052, Hewlett-Packard 9845 und Commodore 3032 besteht. Zwei der Prozeßrechner 330 und ein Tektronix-Rechner sind untereinander und an einen Großrechner TR440 gekoppelt; und einige Kleinrechner sind an andere Siemens-Rechner 330 angeschlossen. Unter Rechnerverbund wird aber nicht allein die Kopplung der Rechner untereinander verstanden, sondern ihre Ausstattung mit einheitlichen Schnittstellen und Bausteinen der Hardware und Software, wodurch im gesamten System eine weitgehende Kompatibilität der Hardware, der Daten und der Programme erreicht und Arbeitszeit und Geldmittel bei den einzelnen Automatisierungsaufgaben gespart werden.

a) Hardware-Kompatibilität: Prozeß- und Meßgeräte ohne standardisierte Schnittstelle (V24 oder IEC-Bus) werden an die Prozeß- und Kleinrechner mit Hilfe des in der PTB entwickelten und im großen Umfang eingesetzten universellen Interface (Ein-/Ausgabe-Steuerung, Lit. s. Ref. 1) angeschlossen. Dieses Interface kann mit digitalen und analogen Ein- und Ausgängen, Alarmeingängen, Zählern, Timern, V24-Schnittstellen und

Sonderplatinen bestückt werden und läßt sich intern programmieren und dadurch fast jedem vorkommenden Prozeß sehr kostengünstig im Vergleich zu anderen Lösungen anpassen. Es entspricht hinsichtlich seiner Ansteuerung der Siemens-Prozeßeinheit 3600, ist jedoch flexibler und vielfältiger bei der Schnittstellenanpassung und schneller bei der Datenübertragung, z.B. im Speicherdirektzugriff, und kann über IEC-Bus oder eine Speicher-Schnittstelle auch an Kleinrechner angeschlossen werden. Software-Routinen für die Ansteuerung der Ein-/Ausgabe-Steuerung durch jeden Rechner, sowie Alarm- und Testprogramme ergänzen das Interface, hierüber wird an anderer Stelle berichtet [1]. Alle Peripheriegeräte wie Blattschreiber, Drucker, Sichtgeräte, Plotter, Disketten- und Kassettenmagnetband-Geräte werden mit V24- oder IEC-Bus-Schnittstelle verwendet, die Ankopplung der Kleinrechner an die Siemens-Prozeßrechner geschieht ebenfalls so [2].

b) Daten-Kompatibilität: Die bei Meßprozessen gewonnenen Daten müssen meistens gespeichert, archiviert und von speziellen Programmen auf dem selben oder einem anderen Rechner verarbeitet werden. Sie werden deshalb auf kompatiblen Datenträgern (Magnetplatten, Disketten, Kassettenmagnetbänder) nebst den dazugehörigen Speichergeräten so abgelegt, daß sie von Programmen in unterschiedlichen Rechnern gelesen werden können, entweder direkt von transportierten Datenträgern oder als mittels Rechnerkopplung überspielte Daten. Dazu gehören eine gewisse Normierung der Daten und Dateien und für die verschiedenen Rechner entsprechende allgemeine Routinen und Dienstprogramme für Lesen, Schreiben, Konvertieren, Editieren, Transferieren über Rechnerkopplung und das Anstoßen von Verarbeitungsprogrammen in einem angeschlossenen Zielrechner. Mit dem hier angesprochenen Aspekt befaßt sich ausführlich ein anderer Vortrag [3].

c) Programm-Kompatibilität: Leider muß für jeden der sehr unterschiedlichen Meßprozesse ein eigenes individuelles Automatisierungs-Programmsystem geschrieben werden. Doch wird die Arbeit sehr erleichtert durch die Verwendung universeller Routinen, Programmbausteine und Dienstprogramme, z.B. für die Bedienung, die Ansteuerung der Prozeßgeräte am Interface, den Datenverkehr und die Graphik; sowie durch die Programmierung in Prozeß-FORTRAN und BASIC. Nur ausnahmsweise wird bei speziellen Softwareteilen Assembler benutzt. Sehr wichtig für die Transparenz ist auch der einheitliche Aufbau des Programmsystems, was gleich näher ausgeführt werden soll. Bei Beachtung der genannten Prinzipien, mit Modellprogrammen als Vorlage und mit einigen Tips können auch relativ ungeübte Experimentatoren ihren eigenen Prozeß automatisieren.

2. Programmstruktur

Ein Programmsystem für die Automatisierung eines allgemeinen Meßprozesses, der simultan zu höchstens zwei bis drei anderen am selben Siemens-Prozeßrechner ablaufen soll, kann erfahrungsgemäß am einfachsten in ein hauptspeicherresidentes Alarmprogramm und ein peripherspeicherresidentes Programm (evtl. auch mehrere) strukturiert werden (s. Abb. 1). Sie synchronisieren und verständigen sich mittels Koordinierungszähler und tauschen Daten über einen globalen COMMON-Bereich aus.

Das Alarmprogramm ist in Assembler geschrieben und ist für jeden Prozeß lediglich durch einige spezifische Parameter und einzelne Alarmfunktionen zu ergänzen [1]). Es koordiniert die Alarme für peripherspeicherresidente Programme und überträgt ihnen die Bearbeitung, führt aber auch kurze zeitkritische Arbeiten selbst aus.

Die übrigen Aufgaben des Prozesses, insbesondere Tätigkeiten nach Bedienungskommandos, erledigt das peripherspeicherresidente

Programm, z.B. das Setzen von Parametern, das Abspeichern, Wiederlesen und Verarbeiten von Meßdaten, das graphische Darstellen von Ergebnissen auf Zeichen- und Sichtgeräten, aber auch eigentliche Prozeßfunktionen wie Initialisieren oder Einlesen von Meßdaten nach Ablauf einer Meßzeit, d.h. umfangreichere Tätigkeiten nach Alarmen, die vom Alarmprogramm delegiert werden. Das Programm kann unter Verwendung vorgefertigter Softwareteile in FORTRAN geschrieben werden. Es besteht aus einem prozeßunabhängigen standardisierten Programmkopf und einem Rumpf mit Unterprogrammen für die prozeßspezifischen Funktionen. Der Rumpf kann segmentiert sein. Der Kopf enthält im wesentlichen einen Bedienungsteil mit Kommando-Interpreter, der Bedienungskommandos formal entschlüsselt und ihre Parameter bereitstellt, sowie einen Verteiler, der das jeweilige kommandoausführende Unterprogramm, evtl. auch ein weiteres Programm ermittelt und aufruft.

Das Programmsystem kann ergänzt werden durch weitere, ähnlich strukturierte peripherspeicherresidente Programme mit oder ohne Standard-Programmkopf oder durch Dienstprogramme. Diese Programme erledigen im allgemeinen die umfangreicheren, längerdauernden oder selteneren Arbeiten. Alle Programme im Laufbereich wechseln im Timesharing-Modus, damit die Simultanarbeit mehrerer Prozesse möglich wird.

3. Standard-Programmkopf

Weil das Entschlüsseln von Bedienungsanweisungen erfahrungsgemäß in einem Prozeßprogramm einen breiten Raum einnimmt, war es zweckmäßig, dafür den Standard-Programmkopf zu erstellen. Seine Struktur ist in Abb. 2 dargestellt.

Der Kopf besitzt eine zentrale Koordinierungsstelle, die einzige des gesamten Programmes, wo auf die Erhöhung eines Koordinierungszählers gewartet wird. Dieser Zähler wird beim Start eingerichtet und kann erhöht werden durch Bedienung oder

z.B. durch das Alarmprogramm, um die Bearbeitung eines Alarms einzuleiten.

Beim Weiterlauf wird geprüft, ob ihn eine Bedienung veranlaßt hat. Wenn ja, wird der Kommando-Interpreter aufgerufen, der die Parameter eines Bedienungskommandos in einem Parameterfeld bereitstellt; wenn nein, werden ein "Alarm"-Codewort und die mit dem Koordinierungszähler übergebenen Daten im Parameterfeld abgelegt. Beim Start werden die Parameter eines Initialisierungskommandos direkt in das Parameterfeld geschrieben.

Nachdem im Parameterfeld die Daten für eine Aufgabe vorliegen, wird der Verteiler aufgerufen, ein Unterprogramm, das ihre Bearbeitung veranlaßt. Das Programm kehrt nach Erledigung der Aufgabe, wenn diese kein Kommando war, zur Koordinierungsstelle zurück; anderenfalls prüft es, ob in dem Bedienungspuffer - dieser wird beim Start gelöscht - noch weitere Kommandos anstehen. Wenn ja, ruft das Programm wieder den Kommando-Interpreter auf; wenn nein, wird vor dem Anlaufen der Koordinierungsstelle ein Bedienungsaufruf über den Koordinierungszähler abgegeben.

Der Verteiler vergleicht den ersten Parameter im Feld, der als Codewort aufgefaßt wird, mit dem Inhalt einer prozeßspezifischen Codewortliste, ermittelt so das Unterprogramm des Programmrumpfes, das die Aufgabe bearbeitet, und ruft es auf. Die Liste ist in Codewortgruppen geordnet, für die jeweils ein Unterprogramm zuständig ist. Falls die Aufgabe ein Kommando ist, kann sie der Verteiler auch an ein anderes Programm übertragen. Er schreibt dazu den Inhalt des Parameterfeldes in ein zweites Feld im globalen COMMON-Bereich, wenn es frei ist, belegt es dadurch und erhöht einen Koordinierungszähler des Bearbeitungsprogrammes.

Ist das zweite Parameterfeld nicht frei oder stellt der Kommando-Interpreter ein syntaktisch falsches Kommando fest,

werden die Abweisung bzw. der Fehler des Kommandos gemeldet, der Bedienungspuffer gelöscht und eine neue Bedienung angefordert.

Der Standard-Programmkopf ist in FORTRAN geschrieben, abgesehen von den Teilen für die Koordinierung, für die Unterprogramme in Assembler benutzt werden [1]. Es ist geplant, den Kopf um eine Monitorfunktion zu erweitern, so daß auch eine vorgefertigte Kommandofolge in einer Datei automatisch abgearbeitet werden kann.

4. Kommando-Interpreter

Der Kommando-Interpreter des Standard-Programmkopfs ist ein in FORTRAN geschriebenes Unterprogramm, das die eingegebene Zeichenfolge eines Bedienungskommandos nach syntaktischen Regeln dekodiert und auf formale Fehler prüft. Die entschlüsselten und in die intern benutzbare Form umgewandelten maximal 15 Parameter eines Kommandos werden jeweils in 4 Zellen des 60 Speicherworte langen Parameterfeldes in einem COMMON-Bereich abgelegt, und zwar je Parameter entweder 8 alphanumerische Zeichen oder in den beiden ersten Zellen eine ganze Zahl (INTEGER*4) als Typkennung und in der dritten und vierten Zelle den Wert als ganze Zahl (INTEGER*4), als 32-Bit-Muster (INTEGER*4) oder als reelle Zahl (REAL*4). Nicht eingegebene Parameter erhalten einen Wert aus 8 Zeichen Zwischenraum ␣. Der erste Parameter ist im allgemeinen das übliche Kommando-Codewort.

Die Syntax der universellen Kommandosprache wurde so allgemein gewählt, daß sie in fast allen Fällen genügend Spielraum für die Definition üblicher sinnvoller Kommandos bietet, andererseits mnemotechnisch doch so geregelt ist, daß möglichst viele Bedienungsfehler schon bei der formalen Prüfung der Kommandos erkannt werden. Sie umfaßt die Syntax der Bedienung der meisten Siemens-Dienstprogramme und soll hier nur in großen Zügen erläutert werden.

Eine Bedienungseingabe umfaßt eine Zeile von maximal 72 Zeichen einschließlich des abschließenden ETX-Zeichens ◊. Sie kann mit dem Anruf :n: des Programms mit der Nummer n beginnen und mehrere durch ; getrennte Kommandos enthalten. Vor ◊ kann ; entfallen. Jedes Kommando besteht aus Zeichenfolgen, die die maximal 15 Parameter darstellen. Sie sind durch Trenner voneinander abgesetzt. Ein Trenner ist eines der Zeichen : , = ␣ oder eine Folge aus lauter ␣ vor einem jener Zeichen. Werden ein Trenner oder das Ende des Kommandos (; oder ◊) erkannt, wenn ein Parameter erwartet wird, so gilt dieser Parameter bzw. gelten alle restlichen als leer, als nicht eingegeben.

Eine Parameterzeichenfolge darf keines der Zeichen eines Trenners enthalten, außer zwischen den einschließenden Zeichen " der Zeichenfolge eines Textparameters. Führende Zeichen ␣ sind redundant. Die Parameterzeichenfolge kann bedeuten:

- eine ganze Zahl, wenn sie aus maximal 9 Ziffern mit oder ohne Vorzeichen + oder - besteht.
- eine reelle Zahl r ($0{,}5\cdot10^{-38} \leq |r| \leq 1{,}5\cdot10^{+38}$), wenn sie in der üblichen Weise dargestellt ist: Mantisse, evtl. mit Vorzeichen und Dezimalpunkt, ggf. Exponent mit E und Vorzeichen, insgesamt maximal 17 Zeichen; der Betrag des Exponenten darf 38 nicht übersteigen.
- ein Bitmuster von 32 bit, wenn sie aus maximal 8 Hexadezimalziffern Ø,...9,A,...F, eingeschlossen in Zeichen ' besteht (Hexaparameter). Die Hexadezimalziffernfolge wird ggf. mit angehängten Ziffern Ø auf 8 Zeichen ergänzt.
- ein Zeichenstring, wenn sie aus beliebig vielen abdruckbaren Zeichen außer " besteht, die in Zeichen " eingeschlossen sind (Textparameter). Je 8 Zeichen bilden einen Parameter; ggf. werden Zeichen ␣ angehängt, damit die Zeichenzahl durch 8 teilbar wird.
- ebenfalls ein Zeichenstring, wenn sie aus beliebig vielen abdruckbaren Zeichen außer denen eines Trenners besteht,

nicht mit ' oder " beginnt und wenn sich unter den ersten 9 Zeichen wenigstens eines befindet, das erkennen läßt, daß die Parameterzeichenfolge keine ganze oder reelle Zahl darstellt (Wortparameter). Wieder werden Zeichen ␣ hinzugefügt, bis die Zeichenanzahl gleich 8n ist (n natürliche Zahl). Der String umfaßt dann n Parameter, Trenner sind hier nicht nötig. Das Kommando-Codewort ist ein Wortparameter.

Ein syntaktischer Fehler wird gemeldet und seine Lage in der eingegebenen Zeichenfolge markiert. Die Eingabe wird gelöscht. Die noch nicht abgearbeiteten Kommandos müssen danach neu eingegeben werden. Bedienungszeichenfolgen, die so aufgebaut sind wie z.B. die für das Standardbedienprogramm des Betriebssystems, sind für den Kommando-Interpreter im allgemeinen fehlerfrei.

Beispiele

Bezeichnungen:

z	ganze Zahl	⊢—⊣	Parameter, allgemein
r	reelle Zahl	|	oder ⊢~⊣ Trenner
h	Hexaparameter	◊	ETX
a	Text- oder Wortparameter	␣	Zwischenraum
⊥	Leerparameter	≀	Kommandoende

```
:4:PETER␣IST␣DOOF◊
   a | a | a ≀
```

```
A=3,B=-Ø.7E-8,C:'F37ØA',,3*="NAME=*";RUN␣11␣6◊
a|z|a|  r  |a|  h  |||a|  a  ≀ a | z | z ≀
```

```
DAMPFSCHIFFAHRT␣␣␣:␣␣=␣"␣KAPITAEN␣";◊
 a     a    ~|||~    a    a ≀≀
```

```
␣/LADE:EG⟨PLSK(Ø),GSB⟩,DIPOS:LB1:PP5;◊
 a |  a   a | a | a | a | a ≀≀
```

Literatur

[1]) Schuster, H.-J.; Weise, K.: Universelle Routinen für Siemens-Prozeßrechner 300 zur Alarmbehandlung in peripherspeicherresidenten Programmen. Bericht der 11. Jahrestagung des Siemens-Prozeßrechner-Anwenderkreises, KFA Jülich 1980 (dort weitere Literatur).

[2]) Weise,K.: Rechnerkopplung Tektronix 4051 - Siemens 330. Bericht der 9. Jahrestagung des Siemens-Prozeßrechner-Anwenderkreises, KfK Karlsruhe 1978, Ber. KfK-2642.

[3]) Plewnia, A.; Lindemann, H.: Dateitransfer im Dialogmodus. Bericht der 11. Jahrestagung des Siemens-Prozeßrechner-Anwenderkreises, KFA Jülich 1980.

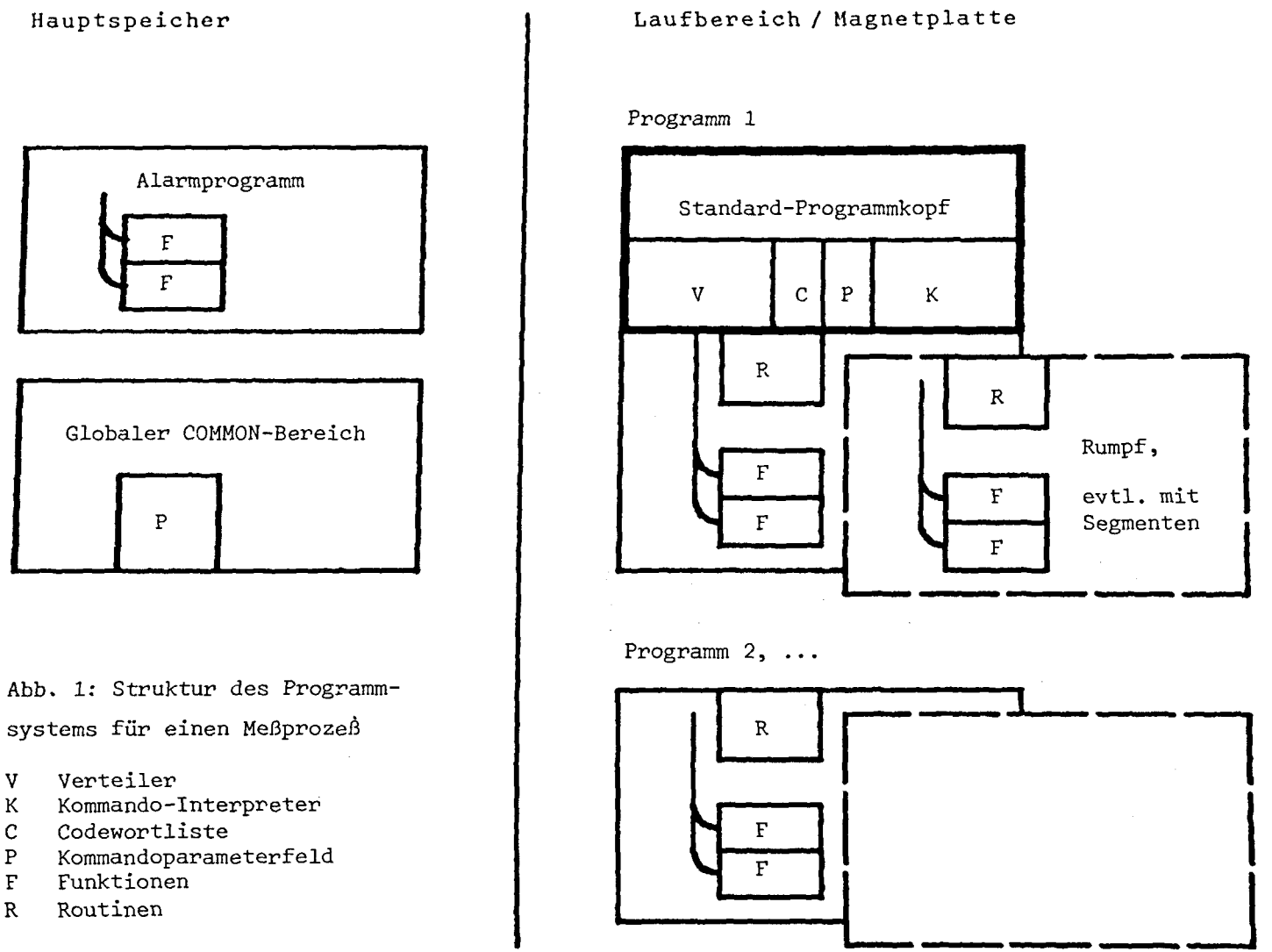
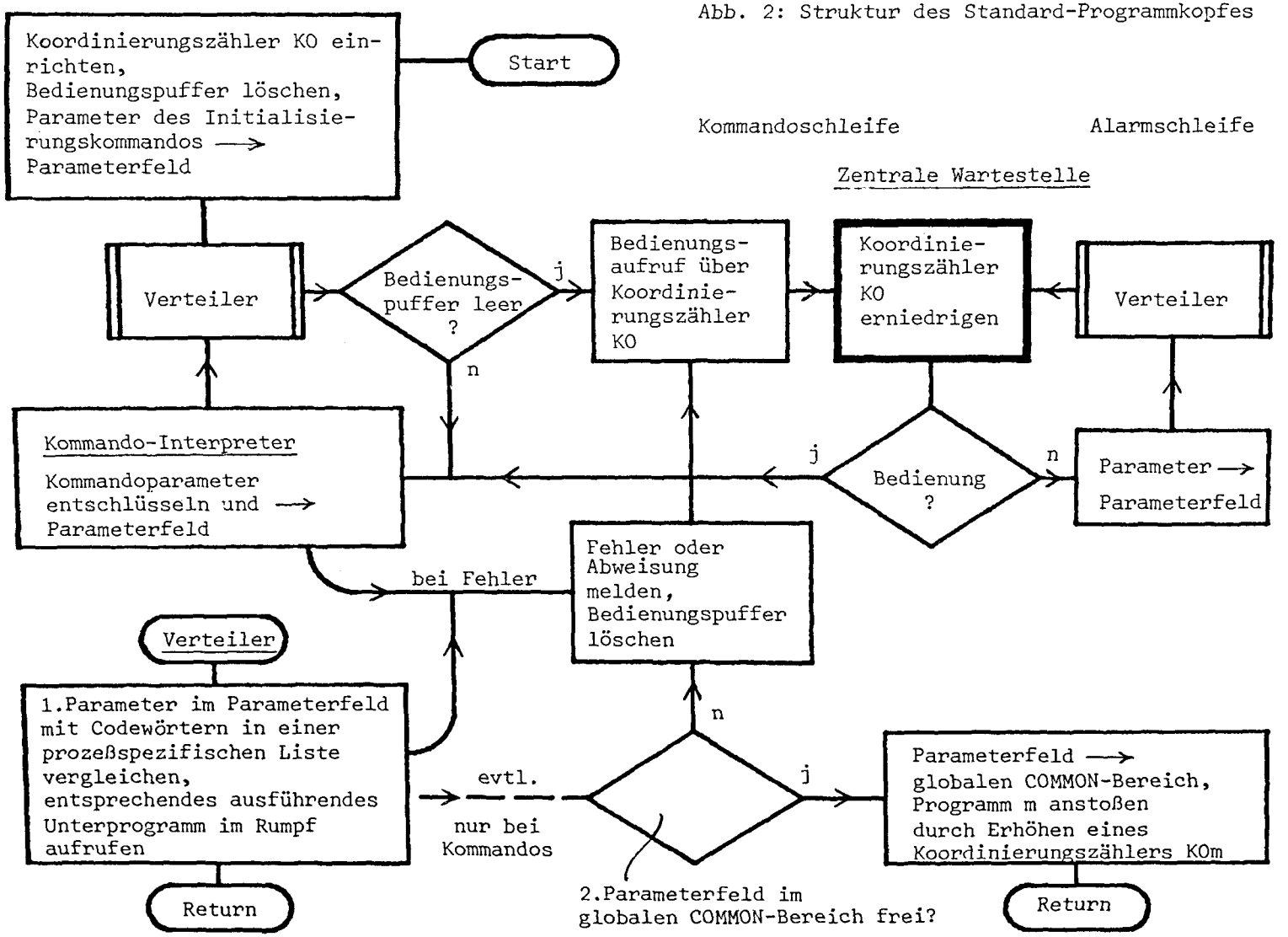


Abb. 1: Struktur des Programmsystems für einen Meßprozeß

V Verteiler
K Kommando-Interpreter
C Codewortliste
P Kommandoparameterfeld
F Funktionen
R Routinen

Abb. 2: Struktur des Standard-Programmkopfes



Universelle Routinen für Siemens-Prozeßrechner 300 zur Alarmbehandlung in peripherspeicherresidenten Programmen

H.-J. Schuster, K. Weise

Physikalisch-Technische Bundesanstalt, Braunschweig

1. Einleitung

Etwa 100 Meß- und Prüfprozesse aus fast allen Fachlaboratorien der Physikalisch-Technischen Bundesanstalt (PTB) sollen mit Hilfe eines Prozeßrechner-Verbundsystems, bestehend aus 17 Prozeßrechnern Siemens 300 und ca. 60 Kleinrechnern (Tektronix 4051, Commodore, Hewlett-Packard), automatisiert werden. Alle dafür erforderlichen und zentral auszuführenden Arbeiten, wie Beschaffung, Wartung, Beratung, Ausbildung, Anschluß von Meßprozessen, werden vom Laboratorium "Zentrale Prozeßdatenverarbeitung" übernommen.

Im Rahmen dieser Arbeiten werden auch allgemein verwendbare Routinen für Siemens-Prozeßrechner entwickelt, durch die der Software-Anschluß der Prozesse erleichtert und rationalisiert wird. Diese allgemeinen Routinen lassen sich funktionell in 3 Klassen gliedern:

- Routinen für die Interface-Ansteuerung. Sie dienen der zentral und peripher initiierten Datenübertragung (Datenalarme) sowie der Behandlung von Organisationsalarmen
- Routinen für die Steuerung und Parameterversorgung der Anwenderprogramme
- Routinen für die Datenübertragung zwischen Rechnern

In dieser Arbeit werden die Routinen der ersten Klasse, also im wesentlichen Routinen zur Alarmbehandlung, vorgestellt.

2. Allgemeine Struktur der Software für die Automatisierung von Meßprozessen mittels Siemens-Prozeßrechnern

Die Automatisierung der Meßprozesse mittels Siemens-Prozeßrechnern erfolgt nach einheitlichen Richtlinien, um dabei möglichst wenig Arbeitszeit und Mittel für Geräte aufwenden zu müssen.
Einige Richtlinien und Anforderungen, die auch die allgemeine Struktur der Software für die Automatisierung der Prozesse beeinflussen, sind diese:

2.1 An jeden Siemens-Prozeßrechner werden etwa 3 simultan laufende Meßprozesse angeschlossen.
2.2 Als Programmiersprache wird grundsätzlich Prozeß-Fortran benutzt. In Ausnahmefällen, wenn Rechenzeit oder Speicherplatz eingespart werden müssen, werden spezielle Routinen auch in der Assembler-Sprache ASS 300 programmiert und zwar in der Regel im Laboratorium "Zentrale Prozeßdatenverarbeitung", weil den Anwendern der Umgang mit der Assemlber-Sprache nicht zugemutet werden soll.
2.3 Prozeßgeräte (z.B. Meßgeräte) ohne standardisierte Schnittstelle (z.B. V24 oder IEC-Bus) werden an die Rechner mit Hilfe des in der PTB entwickelten Interfaces angeschlossen. Dieses Interface (EA-Steuerung) entspricht hinsichtlich seiner Ansteuerung weitgehend der Prozeßeinheit 3600 [1]). Es bietet aber bessere und vielfältigere Möglichkeiten einer flexiblen Schnittstellenanpassung und der Datenübertragung im Speicherdirektzugriff (DMA-Mode).
2.4 Für jeden der an einen Siemens-Rechner angeschlossenen Prozesse sollen Alarmreaktionszeiten von höchstens 500 µs realisiert werden können, unabhängig vom Zustand der simultan laufenden Prozesse.
2.5 Zwischen einem Prozeß und dem Rechner sollen jederzeit Daten mit Raten von mindestens 20.000 Worten/s übertragen werden können, unabhängig vom Zustand der simultan laufenden Prozesse.

Durch diese Richtlinien und Anforderungen ist die allgemeine Struktur der Software für die Prozeßautomatisierung, die in Bild 1 dargestellt ist, bereits weitgehend festgelegt.

Die Software für einen Prozeß besteht in der Regel aus folgenden Teilen:

- Programm 1 mit einem Standardprogrammkopf und dem Unterprogramm EASTKO
- Programm 2 und ggf. weitere Programme
- Alarmprogramm
- Common-Data-Bereich
- Testprogramm

Die eigentliche Prozeßbearbeitung erfolgt, abgesehen von zeitkritischen Maßnahmen, im Programm 1 und wenn mehrere Maßnahmen simultan auszuführen sind, zusätzlich im Programm 2 und evtl. weiteren Programmen.

Diese Bearbeitungsprogramme sind im allgemeinen umfangreich, da sie entsprechend der Richtlinie 2.2 in Prozeß-Fortran geschrieben werden, und müssen außerdem entsprechend Richtlinie 2.1 simultan zu mehreren anderen Programmen (nach Bild 1 insgesamt etwa 7 Stück) laufen. Daher können sie wegen des begrenzten Arbeitsspeicherraumes nur als peripherspeicherresidente Programme (PSP-Programme) geladen und in einem gemeinsamen Laufbereich ablaufen.

PSP-Programme dürfen nicht direkt auf Alarme warten, weil dann der Laufbereich für andere Programme gesperrt ist (Zustandswechselsperre). Daher muß bei PSP-Programmen das Warten auf Alarme über Koordinierungszähler abgewickelt werden. Dazu wird im PSP-Programm ein Koordinierungszähler eingerichtet und dieser erniedrigt, wenn der Wartezustand herbeigeführt werden soll. Das Programm wartet nun, bis der Koordinierungszähler wieder erhöht wird, wobei es jetzt den Laufbereich nicht blockiert. Es wartet dann auch indirekt auf Alarme, wenn dafür gesorgt wird, daß bei ihrem Eintreffen der Koordinierungszähler erhöht wird.

Das Erhöhen von Koordinierungszählern oder Wecken von PSP-Programmen als Reaktion auf eintreffende Alarme übernimmt das hauptspeicherresidente und in Assembler 300 geschriebene Alarmprogramm ALARMj (s. Bild 1 und Abschn. 3). Die Bearbeitung der Alarme erfolgt dann in den geweckten PSP-Programmen (z.B. Programm 1 u. 2 in Bild 1). Diese müssen ggf. erst in den

Laufbereich geladen werden, was die Alarmbearbeitung um ca. 50 ms verzögert.

Das Alarmprogramm kann aber auch zeitkritische Alarmbearbeitungen selbst übernehmen, die dann in der Regel ca. 300 µs nach dem Eintreffen des Alarmes ausgeführt werden. Dadurch wird die Anforderung 2.4 erfüllt.

Ein weiterer standardisierter, in ASS 300 geschriebener Modul ist das Unterprogramm EASTKO (Ein-Ausgabe-Steuerung mit Koordinierungszählern). Es enthält Routinen für die Interface-Ansteuerung einschließlich der Alarmbehandlung mit Koordinierungszählern, die von einem Fortran-Programm aufgerufen werden können. Diese Aufrufe können z.B. auch eine Datenübertragung im DMA-Modus (Blockverkehr) veranlassen, wodurch die Anforderung 2.5 erfüllt wird. Das Unterprogramm EASTKO ist in der Regel ein Teil der PSP-Programme (z.B. Progr. 1 u. 2 in Bild 1) und wird in Abschn. 4 näher beschrieben.

Zum Testen des Interfaces wurde das Programm TESTKO (s. Abschn. 5) entwickelt. Dieses Fortran-Programm gestattet, die Routinen von EASTKO mittels Bedienungsanweisungen aufzurufen. Damit können alle Ein-/Ausgabe-, Alarm- und Koordinierungsfunktionen des Interfaces für Testzwecke ausgelöst werden.

Schließlich ist jedem Prozeß in Bild 1 noch ein hauptspeicherresidenter Common-Data-Bereich (globaler COMMON-Bereich) zugeordnet. Über diesen globalen Bereich werden Daten zwischen allen Programmen ausgetauscht oder Ein-/Ausgabedaten gepuffert.

3. Das Alarmprogramm ALARMj

Das Alarmprogramm hat im wesentlichen die Aufgabe, auf Alarme zu warten, eintreffende Alarme zu interpretieren und über Koordinierungszähler Aufrufe an die zuständigen PSP-Programme weiterzuleiten, die dann ihre Bearbeitung übernehmen. Besonders zeitkritische Alarmbearbeitungen kann das Alarmprogramm auch selbst ausführen.

Beim Entwurf des Alarmprogrammes wurde davon ausgegangen, daß jeder Prozeß über eine oder auch mehrere gekettete EA-Steue-

rungen [1]) des Prozeß-Interfaces an eine eigene EA-Schnittstelle mit der Kanalnummer j angeschlossen wird. Außerdem soll jedem Prozeß auch ein eigenes Alarmprogramm ALARMj zugeordnet werden, das mit der Kanalnummer j des zugehörigen Prozesses gekennzeichnet ist. Die Alarmprogramme verschiedener Prozesse unterscheiden sich durch die Kanalnummer, Koordinierungszählernamen, Alarmmasken und ggf. Alarmbearbeitungsroutinen. Aus diesem Grunde ist das Alarmprogramm für jeden Prozeß speziell zu erstellen. Dazu sind in ein vorhandenes Musteralarmprogramm die prozeßspezifischen Parameter einzutragen und das entstandene aktualisierte Assemblerprogramm zu übersetzen.

Im folgenden wird ALARMj an Hand des Ablaufdiagrammes in Bild 2 beschrieben. Nach dem Start von ALARMj wird ein für das Mehrfachwarten des Alarmprogrammes (es wartet auf Bedienung und Alarme) erforderlicher Koordinierungszähler des Namens Aj eingerichtet. Weiterhin werden Organisationsalarme (Primäralarme) mit der Alarmnummer ji und eine Bedienung angemeldet, wobei in den GEDA-Blöcken zu diesen beiden Aufrufen Aj als zu erhöhender Koordinierungszähler eingetragen wird [3]). Danach wird das Programm durch Erniedrigen von Aj in den Wartezustand versetzt und erst durch das Eintreffen eines Alarmes ji oder einer Bedienungsanweisung vom Eingabegerät wieder fortgesetzt. Es wird gleich fortgesetzt, wenn bereits eines der genannten Ereignisse eingetroffen war. Im Falle der Fortsetzung durch eine Bedienungsanweisung wird der Eingabestring, der die auszuführende Bedienungsmaßnahme spezifiziert, abgefragt. Heißt er "ENDE" ist das Alarmprogramm zu beenden. Dazu werden der Koordinierungszähler Aj gelöscht, der Alarm ji abgemeldet und der Ende-Aufruf gegeben.

Außer dem Eingabestring "ENDE" könnten noch andere Eingabestrings vereinbart sein. Sie würden durch entsprechende zusätzliche Abfragen erkannt und die dadurch spezifizierten Bedienungsmaßnahmen durchgeführt. Dies sei durch den gestrichelten Programmweg in Bild 2 dargestellt. Nach der Ausführung der Bedienungsanweisung wird eine neue Bedienung angemeldet und das Programm wieder in den Wartezustand versetzt.

Im Falle der Fortsetzung des Programmes durch einen Organisa-

tionsalarm (Primäralarm) wird zunächst der 16bit-Alarmspeicher (s. Ref. 1, S. 19) löschend abgefragt und das eingezogene Alarmwort in der Zelle ALSPEI (Alarmspeicher) abgelegt. Jedes gesetzte Bit in ALSPEI weist dann einen eingetroffenen Sekundäralarm aus. Damit ein Alarmprogramm mehrere PSP-Programme bedienen kann, werden die Sekundäralarme gruppenweise weiterbearbeitet. Jedem zu bedienenden PSP-Programm wird eine Alarmgruppe zugeteilt und außerdem ein Koordinierungszähler zugeordnet. Die Alarmgruppen werden durch Masken (prozeßspezifischer Parameter) festgelegt. In der Zelle MASKEi sind alle Bits gesetzt, die den Sekundäralarmen aus der Gruppe i zugeordnet sind.

Die Bearbeitung der Sekundäralarme in der Zelle ALSPEI beginnt mit der Prüfung, ob Alarme der ersten Gruppe 1 eingetroffen sind bzw. die UND-Verknüpfung ALSPEI mit MASKE1 (Alarmgruppenwort) ungleich Null ist (s. Bild 2). Wenn ja, wird der dieser Gruppe zugeordnete Koordinierungszähler erhöht und damit das zuständige PSP-Programm geweckt. Außerdem werden über R6 die eingetroffenen Sekundäralarme der Gruppe in das PSP-Programm übergeben.

Nach der Abfertigung der Sekundäralarme der Gruppe 1 werden auf gleiche Weise die Alarme der Gruppen 2, 3 u.s.w., falls diese im Alarmprogramm vorgesehen sind, abgefertigt. Danach wird das Programm über einen Sprung nach B (s. Bild 2) wieder für die Bearbeitung eines weiteren Primäralarmes oder einer Bedienungsanweisung vorbereitet.

4. Das Unterprogramm EASTKO

Das Unterprogramm EASTKO liefert Funktionen zur Interface-Ansteuerung, die von einem Fortran-Programm aufgerufen werden können.

Die Funktionen des EASTKO sind durch CALL EASTKO (Par. 1, Par. 2, Par. 3, Par. 4) in einem Fortran-Programm aufzurufen. Der Parameter 1 spezifiziert die Funktion, über die Parameter 2 und 3 werden Werte und über Parameter 4 Adressen in Form eines Variablennamens übergeben. Alle Parameter müssen vom Typ INTEGER*2 sein. Im folgenden werden die Routinen des

EASTKO und ihre Funktionen genannt:

- Festlegung der Kanalnummer:
  CALL EASTKO (Ø, Kanalnummer, irrelevant, irrelevant)
  Alle Ein-/Ausgabefunktionen werden automatisch über den Kanal abgewickelt, dessen Nummer zuletzt festgelegt wurde.

- Digitale Eingabe:
  CALL EASTKO (1, Anschluß-Adresse, irr., Name der Zelle in die eingeschrieben wird)

- Digitale Ausgabe:
  CALL EASTKO (2, Anschluß-Adresse, auszugebender Wert, irr.)

- Einrichten des Koordinierungszählers für die Alarmbearbeitung:
  CALL EASTKO (3, Name des Koordinierungszählers, irr., irr.)

- Löschen des Koordinierungszählers für die Alarmbearbeitung:
  CALL EASTKO (4, Name des Koordinierungszählers, irr., irr.)

- Warten auf Sekundäralarme (s. Abschn. 3):
  CALL EASTKO (5, Maske der Sekundäralarmgruppe, irr., Name der Zelle für die aktuellen Sekundäralarme)
  Es wird auf die mit der Maske in Parameter 2 angegebenen Sekundäralarme gewartet.
  Ein gesetztes i-tes Bit in der Maske bedeutet, daß auf den i-ten Sekundäralarm gewartet wird. In die Speicherzelle, deren Name in Parameter 4 steht, wird die Nummer des Sekundäralarmes als Zahl übergeben, aufgrund dessen das Programm fortgesetzt wurde.

- EA-Schnittstelle für Dateneingabe im Blockverkehr vorbereiten:
  CALL EASTKO (6, Länge des Einspeicherbereichs, irr., Name der 1. Zelle des Speicherbereichs)

- EA-Schnittstelle für die Dateneingabe im Inkrement-Mode vorbereiten:
  CALL EASTKO (7, irr., irr., Speicherschutzzelle (Variablenname)

- EA-Schnittstelle für die Datenausgabe im Blockverkehr vorbereiten:
  CALL EASTKO (8, Länge des Ausgabebereichs, irr., Name der 1. Zelle des Ausgabebereichs)

- Koordinierungszähler erhöhen (Alarmsimulation)
  CALL EASTKO (9, Name des Koordinierungszählers, Alarmwort, irr.)

- Bedienung anmelden über Koordinierungszähler:
  CALL EASTKO (10, Name des Koordinierungszählers, Alarmwort,
  Name der 1. Zelle des Bedienungspuffers)
  Eine eintreffende Bedienungsanweisung wird durch den Sekundäralarm, der im Alarmwort spezifiziert ist, gemeldet. Auf Bedienungsanweisungen kann somit wie auf Sekundäralarme gewartet werden.

5. Das Testprogramm TESTKO

Mit Hilfe des Testprogrammes TESTKO können alle Interface-Funktionen des EASTKO durch Bedienungsanweisungen auf einem Eingabegerät für Testzwecke ausgelöst werden.
Das Programm wartet auf Bedienung, interpretiert eintreffende Bedienungsanweisungen und bearbeitet sie durch entsprechende Aufrufe an das Unterprogramm EASTKO. TESTKO enthält EASTKO als Unterprogramm.

Die Bedienungsanweisungen sind formatgebunden und bestehen aus 4 alphanumerischen Zeichen, die die auszulösende Funktion definieren und 4 Hexadezimalziffern zur Parameterübermittlung. Im folgenden werden die möglichen Bedienungsanweisungen und ihre Funktionen genannt:

- KANL NNNN: Die Kanalnummer NNNN wird festgelegt. Alle folgenden Anweisungen werden über diese Kanalnummer abgewickelt
- EING NNNN: Das Bitmuster am Digitaleingang mit der Adresse NNNN wird eingelesen und in eine Pufferzelle geschrieben
- AUSG NNNN: Der Wert in der Pufferzelle wird über den Digitalausgang mit der Adresse NNNN ausgegeben
- EGPE NNNN: Das Bitmuster am Digitaleingang mit der Adresse NNNN wird periodisch in die Pufferzelle geschrieben
- AGPE NNNN: Der Wert in der Pufferzelle wird periodisch über den Digitalausgang mit der Adresse NNNN ausgegeben
- LADZ NNNN: Der Hexadezimalwert NNNN wird in die Pufferzelle geladen
- KOji XXXX: Der Koordinierungszähler ji wird eingerichtet
- KLji XXXX: Der Koordinierungszähler ji wird gelöscht

- WART NNNN: Es wird auf die in der Maske NNNN eingetragenen Sekundäralarme gewartet

- ALji NNNN: Die im Alarmwort NNNN angegebenen Sekundäralarme der Kanalnummer j und der Alarmgruppe i (s. Abschn. 3) werden simuliert

- EGBL XXXX: Die EA-Schnittstelle, deren Kanalnummer zuletzt eingegeben wurde, wird für die Eingabe von 10 Werten im Blockmodus vorbereitet. Die Werte werden in ein Pufferfeld von 10 Zellen geschrieben, daß durch die nächste Anweisung auf dem Bedienungsgerät ausgegeben werden kann

- LESF XXXX: Die 10 16-bit Werte des Pufferfeldes werden hexadezimal auf dem Bedienungsgerät ausgegeben

- EGIN XXXX: Die EA-Schnittstelle, deren Kanalnummer zuletzt eingegeben wurde, wird für die Eingabe im Zählmodus (Inkrement-Modus) vorbereitet

- AGBL XXXX: Die EA-Schnittstelle, deren Kanalnummer zuletzt eingegeben wurde, wird für die Ausgabe im Blockmodus der 10 Werte des Pufferfeldes, das durch die nächste Anweisung geladen werden kann, vorbereitet

- LADF NNNN: Die 10 Zellen des Pufferfeldes werden mit dem im Parameter hexadezimal angegebenen Wert geladen.

Literatur

[1] Schuster, H.-J.: Funktionen und Analyse der Schaltung einer Prozeßdaten-Ein-/Ausgabe-Steuerung für Siemens-Prozeßrechner des Systems 300/16-Bit und Rechner mit IEC-Bus, Physikalisch-Technische Bundesanstalt Braunschweig, Ber. PTB-FMRB-75, (1979)

[2] Weise, K., Rybarsch, R., Schuster, H.-J.: Allgemeine Struktur von Programmsystemen für Meßaufgaben; Programmbausteine und Kommando-Interpreter. Bericht der 11. Jahrestagung des Siemens-Prozeßrechner-Anwenderkreises, KFA Jülich 1980

[3] Siemens AG, ORG 330 K, Beschreibung P 71 100-B 0 330-X-X-35

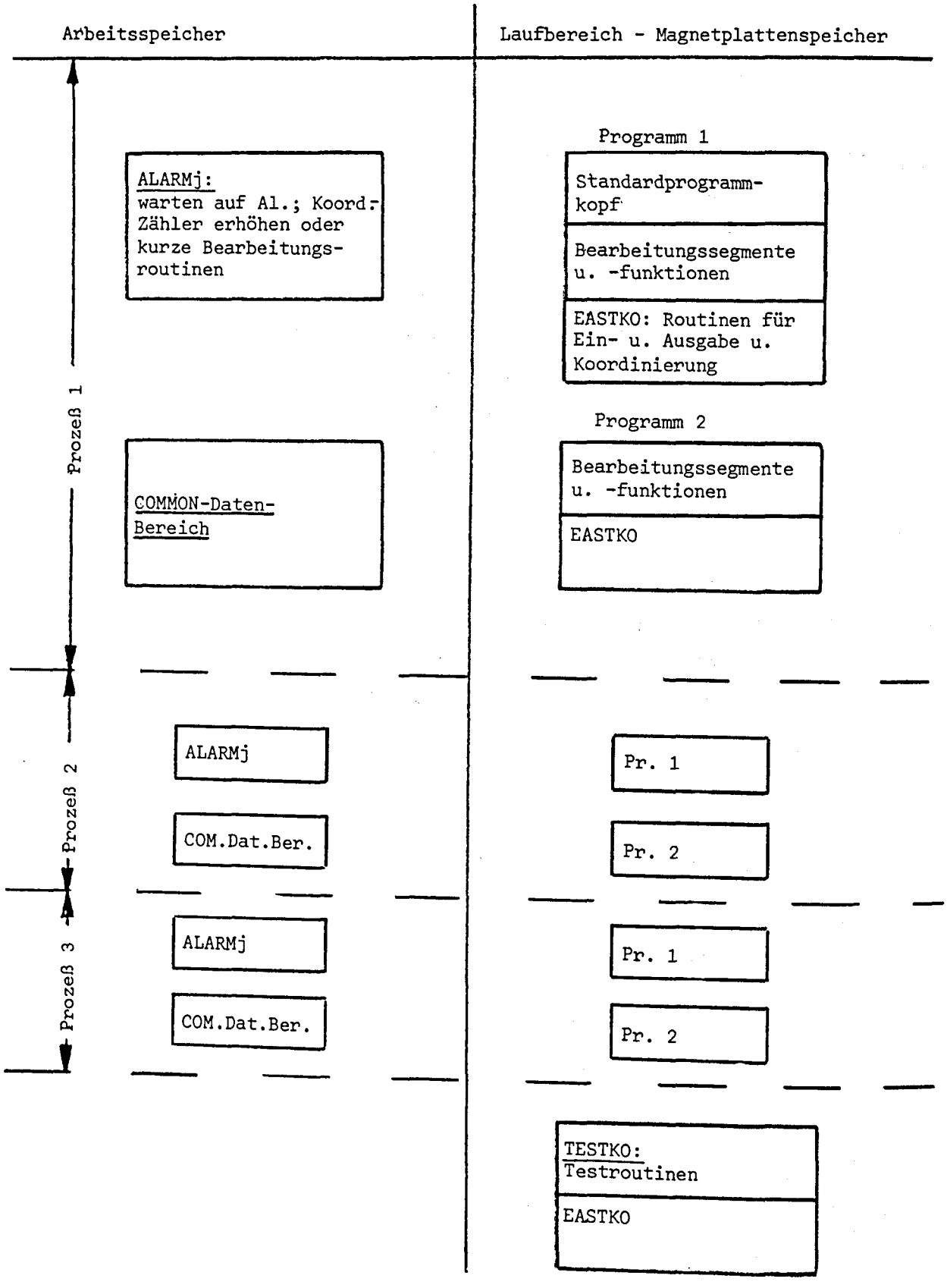


Bild 1: Allgemeine Struktur der Prozeßprogrammsysteme

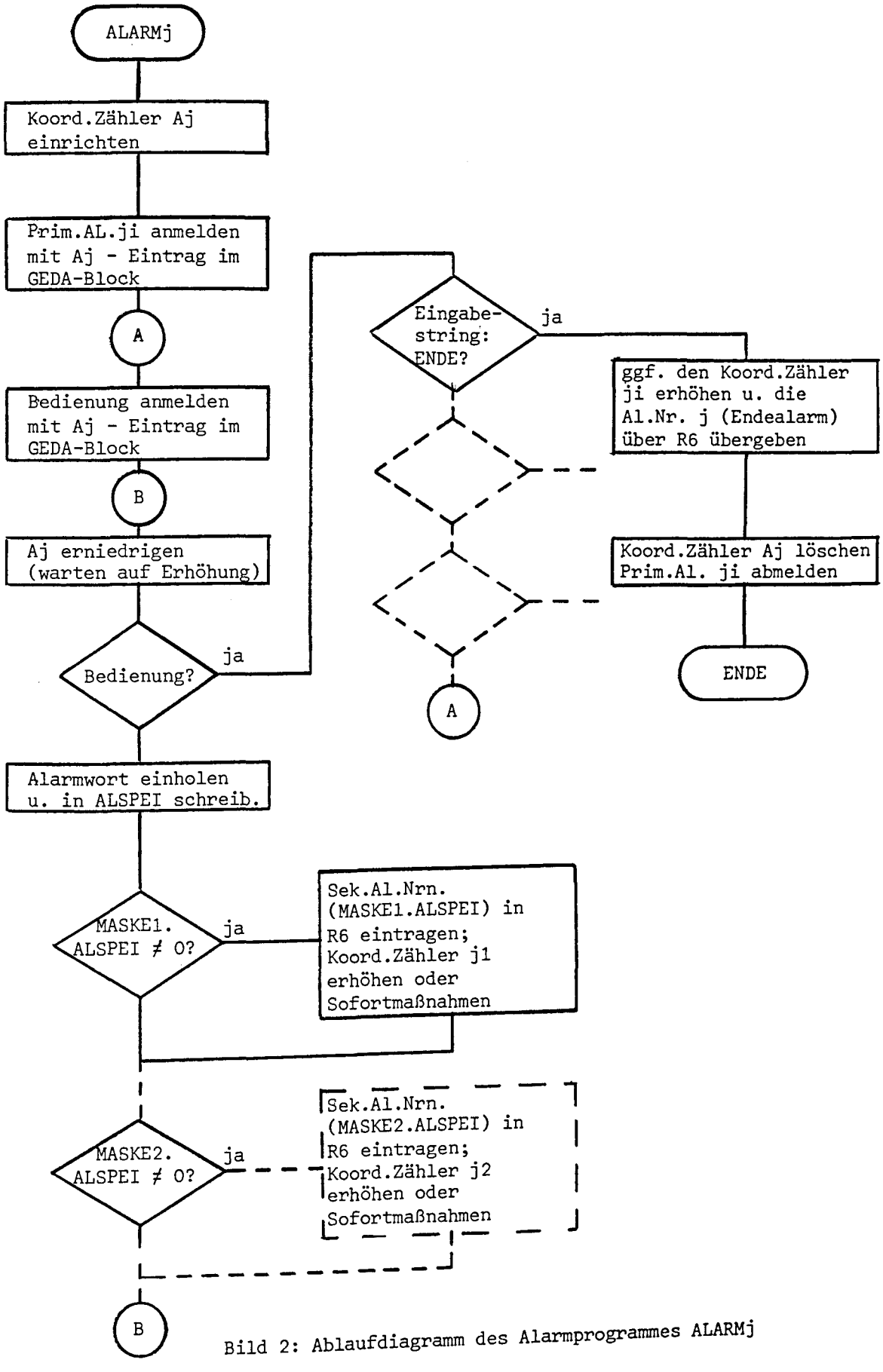


Bild 2: Ablaufdiagramm des Alarmprogrammes ALARMj

Dateitransfer im Dialogmodus

A. Plewnia, H. Lindemann

Physikalisch-Technische Bundesanstalt, Braunschweig

1. Aufgabenstellung

Die Rechenanlagen in der Physikalisch-Technischen Bundesanstalt (PTB) in Braunschweig lassen sich in drei Ebenen einordnen. Die oberste Ebene bildet der Großrechner TR440. Als mittlere Ebene können u.a. die Prozeßrechner Siemens 330 und R30 angesehen werden. Die Rechner 330 sind teilweise durch ein Rechnerkoppelsystem miteinander und mit dem Großrechner verbunden, letzteres im Dialogbetrieb durch Simulation von TR440-Terminals. Die unterste Ebene wird gebildet durch eine größere Anzahl von Kleinrechnern (Tektronix, Commodore, Hewlett-Packard). Diese Rechner haben in den letzten Jahren eine derartige Leistungsfähigkeit erreicht, daß ein großer Teil der Meß- und Prüfaufgaben mit ihnen durchgeführt werden kann. Als einheitlicher externer Datenträger für diese Rechner wird in der PTB die 3M-Kassette verwendet, die auch in der integrierten Bandstation der Tektronix-Rechner 4051/4052 benutzt wird.

Die Aufgaben der Prozeßrechner und der Kleinrechner bestehen vor allem in der Erfassung der bei zahlreichen Meßprozessen anfallenden Meßdaten. Diese Prozesse sind von sehr großer Vielfalt, sowohl was die Datenmenge und Datenrate als auch was die Auswertung der gewonnenen Daten betrifft. Dementsprechend werden die verschiedenen Rechner für diese Aufgaben eingesetzt. Während nun in manchen Fällen der Rechner, der die Daten aufnimmt, auch selbst die vollständige Auswertung durchführen kann, steht häufig die erforderliche Rechenleistung, Speichergröße, Peripherieausstattung und Software nur auf einem Rechner einer höheren Ebene zur Verfügung. Eine Datenübertragungsmöglichkeit zwischen Rechnern verschiedener Ebenen ist daher notwendig.

Diese Übertragungsmöglichkeit muß vor allem sicher und benutzerfreundlich sein. An die Übertragungsgeschwindigkeit brauchen dagegen weniger hohe Anforderungen gestellt zu werden, da es sich nur um die nicht zeitkritische Weiterverarbeitung der erfaßten und gegebenenfalls schon vorbearbeiteten Daten handelt. Unter diesen Voraussetzungen bietet es sich an, die vorhandene Dialog-Verbindung von den Rechnern Siemens 330 zum Großrechner TR440 so auszubauen, daß auch Dateien übertragen werden können. Der gleiche Weg kann bei der Verbindung der Kleinrechner zu den Prozeßrechnern beschritten werden, denn da die Tektronix-Rechner als Bediengerät für Siemens 330 verwendet werden können, steht auch hier eine Dialog-Verbindung zur Verfügung, die für den Dateitransfer geeignet ist. In beiden Fällen wird prinzipiell das gleiche Verfahren verwendet: ein Programm im kleineren Rechner führt einen Dialog mit dem Editor des größeren Rechners und macht sich dadurch dessen Dateiorganisation zugänglich. Auf diese Weise können Dateien in beiden Rechnern bearbeitet und in beide Richtungen übertragen werden. Die Benutzung des Editors erspart die Erstellung einer speziellen Software auf dem Rechner der oberen Ebene.

2. Dateitransfer Tektronix 4051/4052 - Siemens 330

Für einen Dateitransfer zwischen einem Kleinrechner Tektronix 4051/4052 und einem Prozeßrechner Siemens 330 wird auf letzterem der Editor EDIS P 300 durch Bedienung des Prozeßrechners vom Kleinrechner her im Terminal-Betriebsmodus gestartet. Ein spezielles BASIC-Programm im Kleinrechner kommuniziert über eine V-24-Schnittstelle mit dem Editor im Prozeßrechner und erzeugt und übermittelt die Bedienungsstrings an den Editor, die dieses Programm zur Ein- bzw. Ausgabe der Datei oder des Bibliothekselementes benötigt. Dadurch beschränkt sich die Aufgabe des Operateurs auf den Start des Editors im Terminal-Betriebsmodus des Kleinrechners, die Einschaltung des BASIC-Betriebsmodus sowie die Wahl der Übertragungsrichtung. Die

letzten beiden Operationen werden jeweils durch eine Taste des Kleinrechners ausgelöst, was zum Komfort des Dateitransfers beiträgt.

Bei der Übertragung vom Kleinrechner an den Prozeßrechner werden die Daten zeilenweise von der 3M-Kassette im Tektronix-Rechner gelesen und an den Editor übergeben, der die Zeilen auf dem Plattenspeicher des Prozeßrechners in der gewünschten Datei bzw. dem Bibliothekselement ablegt. Für den Editor unverträgliche Datenformate können vor der Übertragung durch eine geeignete BASIC-Subroutine des Dialogprogramms an die Editor-Konventionen angepaßt werden. Die Zeilen werden dem Editor im "hang before"-Modus übergeben. Dieser Modus wird nach der Übertragung einer jeden Zeile abgebrochen, um den Editor zu einer entsprechenden Meldung zu veranlassen, die vom Kleinrechner als Quittung eingelesen und geprüft wird.

Bei der Übertragung in umgekehrter Richtung vom Prozeßrechner zum Kleinrechner ruft das Dialogprogramm in diesem vom Editor im "next line"-Modus eine Zeile ab, die auf die 3M-Kassette übertragen wird. Gegebenenfalls kann auch hier vor der Abspeicherung auf Kassette eine Datenumformatierung erfolgen. Die bei diesem Modus vom Editor im Anschluß an die Ausgabe der Zeile erzeugte Meldung liest das Dialogprogramm ein und prüft sie.

Durch die Quittierung jeder übertragenen Zeile durch den Editor wird eine sehr große Datensicherheit erzielt. Die mit diesem Verfahren auf dem Plattenspeicher des Prozeßrechners erzeugten Dateien bzw. Bibliothekselemente können mit beliebigen Programmen weiterverarbeitet werden.

3. Dateitransfer Siemens 330 - TR440

Die Rechner Siemens 330 sind untereinander teilweise durch das Rechner-Kommunikations-Programm-System RKP 300 verbunden. Das

daran angeschlossene Subsystem DIALOG TR440 simuliert TR440-Terminals und ermöglicht so eine Dialogverbindung mit dem Großrechner TR440. Unter Benutzung der Makros dieser Systeme wurde das Programm FT440 (File Transfer TR440) erstellt, welches die Möglichkeit des Dialogs um die des Dateitransfers in beide Richtungen erweitert.

FT440 stellt nach dem Start zunächst die Verbindung mit dem Großrechner her und wartet dann auf Bedienung. Wenn der eingegebene Bedientext nicht mit dem Zeichen $ beginnt, wird er unverändert an den TR440 übergeben. Dessen Antwort wird ebenso unverändert auf dem Bediengerät ausgegeben und eine neue Bedienung wird erwartet. Es wird also ein normaler Dialog geführt. Bedientexte, die mit $ beginnen, sind Anweisungen an das Programm FT440. Sie steuern den Dateitransfer. Dabei wird vorausgesetzt, daß das Partnerprogramm im TR440 dessen Editor ist. Dieser muß also vor dem ersten Dateitransferbefehl aktiviert worden sein. Die Dateitransferbefehle sind in ihrer Form den entsprechenden DIPOS-Anweisungen angeglichen. Die Anweisung $ED<PLSK(1)>ZZZ bewirkt, daß die angegebene Datei ZZZ mittels BIBEAS gelesen und blockweise, jeweils mit einem Editor-Befehl als Vorspann an den TR440 übergeben wird. Der Editor speichert die übertragenen Blöcke fortlaufend in der von ihm bearbeiteten Datei und sendet eine Quittung zurück. Die Quittung nach dem letzten übertragenen Block wird auf dem Bediengerät ausgegeben und zeigt an, daß der Dialog fortgesetzt werden kann.

Die Anweisung $AD<PLSK(1)>AAA bewirkt, daß die angegebene Datei AAA auf dem Rechner 330 eröffnet wird. Als nächstes wird die Eingabe eines Editor-Befehls erwartet, der Sätze aus einer TR440-Datei auslistet. Diese Sätze werden nicht wie im normalen Dialog auf dem Bediengerät ausgegeben, sondern mittels BIBEAS in die angegebene Datei geschrieben. Die Vollzugsmeldung des Editors wird wiederum auf dem Bediengerät protokolliert und der Dialog kann fortgesetzt werden. Die Dateitransferbefehle können formal so betrachtet werden, als wären sie Erweiterungen

des TR440-Editors, mit denen Dateien und Bibliothekselemente des Rechners 330 angesprochen werden können.

Das Programm FT440 wird beendet durch die Anweisung $STOP. Etwa noch in Bearbeitung befindliche Dateien werden abgeschlossen und die Verbindung zum TR440 wird aufgelöst. Dabei wird vorausgesetzt, daß der Dialog zuvor ordnungsgemäß beendet worden ist.

Die Fehlerbehandlung beschränkt sich darauf, fehlerhafte Transferkommandos zurückzuweisen und die Meldungen von BIBEAS und vom TR440 durchzureichen. Der Dialogbetrieb ermöglicht eine zweckentsprechende Reaktion des Benutzers. Fehler in der Rechnerkopplung führen zum Programmabbruch.

Die Bedienungseingabe und der Verkehr mit dem TR440 werden durch einen Koordinierungszähler gesteuert, so daß auch unangeforderte Meldungen des Großrechners angenommen werden können und das Programm stets bedienbar bleibt.

Beispiel für einen Dialog mit Dateiübertragung

```
/STRT:P32;                                          FT440 starten

/J!
:32:#XBG,BEN=709                                    Dialog anmelden

AUFTRAG 0090  06.05.80  15:39 192215  (1924.01)  0.20  I28-3

GIB KOMMANDOS
:
#LFDATEI,TAG.IN330  #LFANMELDE,TAG.OUT330           Datei IN330 einrichten
KREIERT: IN330(0001.00)  KAT: TAG                   und OUT330 anmelden
ANGEMELDET: OUT330(0001.00)  KAT: TAG

GIB KOMMANDOS
:
#EDIERE,IN330                                       Editor auf IN330 einstellen
DATEI ODER BEREICH IST LEER
:
10=  INHALT VON D<PLSK(1)>ZZZ                       1. Zeile schreiben
:
$ED<PLSK(1)>ZZZ                                     Datei ZZZ von 330 auf TR440
:                                                   übertragen in IN330
+1=  INHALT VON Q<PLSK,QSB>XXX                      weitere Zeile anhängen
:
$EQ<PLSK,QSB>XXX                                    Datei dazuschreiben
:
+1= --- ENDE ---                                    Endezeile anhängen
:
ANFANG,ENDE                                         Inhalt von IN330
10=  INHALT VON D<PLSK(1)>ZZZ                       protokollieren
20=ZZZ - ZEILE 1
30=ZZZ - ZEILE 2
40=ZZZ - ZEILE 3
50=  INHALT VON Q<PLSK,QSB>XXX
60=XXX - ZEILE 1
70=XXX - ZEILE 2
80=XXX - ZEILE 3
90= --- ENDE ---
:
EDIERE OUT330                                       Editor auf OUT330 einstellen
:
$AQ<PLSK(1),QSB>AAA                                 Datei AAA auf 330 einrichten
ANFANG,ENDE                                         Inhalt von OUT330 nach AAA
:                                                   übertragen
STOP                                                Editor beenden

ENDE B&EDITOR (7709.05)  1.08

GIB KOMMANDOS
:
#XEN
##SAS*R10*KONSOLE FREI                              Dialog abmelden
$STOP
FT440 ENDE                                          FT440 beenden
```

G. Bonn, P. Heine
Fraunhofer - Institut
für Informations- und Datenverarbeitung
IITB
Karlsruhe

PEARL-Programmierung auf der SIEMENS 310

Ein PEARL-Betriebssystem

Zusammenfassung:

Verschiedene Lösungswege, ein PEARL-Betriebssystem auf einer S31Ø zu implementieren, werden diskutiert. Die IITB-Lösung wird vorgestellt. Sie läßt das PEARL-Betriebssystem als Anwenderprogramm unter dem ORG 310 ablaufen. Dabei werden nur die Standard-E/A-Funktionen des ORG 310 benutzt.

Das PEARL-Betriebssystem enthält neben den Grundfunktionen ein reentrantfähiges Laufzeitsystem mit dynamischer Speicherverwaltung. Die Schnittstellen PEARL-Anwenderprogramm - PEARL-Betriebssystem - ORG 310 werden diskutiert. Ferner wird näher auf die Koordinierung zwischen dem PEARL-E/A-System und dem ORG 310 eingegangen.

INHALTSVERZEICHNIS

1. Entwicklungshistorie des PEARL-Betriebssystems (PBS1Ø)

Im Laufe der Entwicklung der Systemprogramme für das RDC-Mehrrechnersystem *) wurde 1978/79 ein PEARL-Betriebssystem (PBS) für den RDC-Rechner entwickelt. Der Funktionsumfang des PBS ist durch die echtzeitspezifischen Sprachelemente des PEARL-subsets der Fa. WERUM *) festgelegt, welcher über den Sprachumfang von Basis-PEARL hinausgeht.

Er beinhaltet insbesondere

- Task-Verwaltung
- Zeit- und Interrupteinplanungen
- Synchronisation durch SEMA- und BOLT-Variablen
- SIGNAL-Behandlung
- Prozeß E/A
- Standard E/A

Ferner kooperiert das PBS mit einem Netzwerkbetriebssystem, das den Funktionsumfang der PEARL-Erweiterungen für Mehrrechnersysteme unterstützt.

Die durchweg positiven Erfahrungen, die mit der ersten Pilotimplementierung in der Stahlindustrie gesammelt wurden, führten zu dem Entschluß, auch die Programmerzeugungsstation und die EAF-Station (beides Rechner vom Typ S 310 K) PEARL-fähig aufzurüsten, d.h. den Funktionsumfang des S 310-Betriebssystems durch PEARL-Funktionen zu erweitern. Um den dafür erforderlichen Aufwand möglichst gering zu halten, war die Randbedingung, weitgehend Teile des vorhandenen PBS und des ORG 310 zu benutzen. Vom ORG 310 sollten insbesondere die Teile für die Standard E/A benutzt werden, welche als wesentlichste Ergänzung zum PBS benötigt wurden.

Im folgenden werden 4 Erweiterungsalternativen des ORG 310 auf PEARL-Funktionsfähigkeit kurz diskutiert:

*) vgl. Beitrag von L. Lorenz zu dieser Tagung

2. Erweiterungsalternativen des ORG 310

a) Modifikation der ORG-Moduln

Damit ist gemeint, den ORG-Funktionsumfang durch internes "Aufbohren" der Module der Quellsprache Modul Bibliothek (MOBI) auf PEARL-Niveau anzuheben. Gegen diese Möglichkeit sprechen im wesentlichen folgende Gründe:

- Die ORG-Taskverwaltung ist auf Hardwaremultiprogramming der S 310 abgestimmt (Anzahl der Tasks <10). Daten zur Taskverwaltung sind im ORG-Bereich und nicht im Taskbereich (Taskkontrollblock) abgespeichert.
- Der Funktionsumfang des ORG bietet insbesondere bezüglich der Zeit-Einplanungen nur geringe Möglichkeiten.
- Die Programmpflege des ORG durch Siemens ist nicht mehr gegeben.

b) Einfügen von Anwenderorganisationsmoduln

Der Generiervorgang mit MOBI und SYGE10 bietet die Möglichkeit, vom Anwender geschriebene Betriebssystemmoduln (hier: PEARL-Zusätze) in das ORG einzubinden. Die eigentlichen ORG-Moduln und die Aufrufstruktur (R7 : US 12) bleiben unverändert. PEARL-BS-Aufrufe führen zunächst in einen Anwendermodul (modif. PBS), der die Dation- und Taskverwaltung durchführt. Zum eigentlichen Datentransfer werden ORG-Routinen benutzt, die über ORG interne Aufrufe (ORG-Modul OINA) angesprungen werden.

Diese Alternative scheidet jedoch letztlich dadurch aus, daß OINA-Aufrufe für Dateiorganisation nicht möglich sind.

c) Verwendung der E/A-Treiber als Unterprogramme

Es besteht die Möglichkeit, aus der MOBI eine "stand-alone"-Version zu generieren, in der ein einziges "Anwenderprogramm" (modifiziertes PBS) E/A-Verkehr über ORG-Unterprogrammaufrufe abwickelt.

Diese strukturell eleganteste Lösung hat den großen Nachteil, daß neben den PEARL-Anwenderprogrammen keine weiteren (nicht PEARL-Programme) mehr simultan ablaufen können.

d) PEARL-BS als ORG-Anwenderprogramm

Bei dieser Lösung bleibt das ORG unverändert (kein spezieller Generiervorgang). Damit sind bestehende Programmsysteme simultan zu PEARL-Systemen ablauffähig. Ein modifiziertes PBS, im weiteren mit PBS10 bezeichnet, wickelt den E/A-Verkehr über die ORG-Anwenderschnittstelle (R7 : US 12) ab.

Die Alternative d) wurde für die Implementation ausgewählt. Ihr großer Vorteil ist, daß nicht nur das ORG unverändert bleibt, sondern daß auch am bestehenden PBS nur geringfügige Änderungen vorgenommen werden müssen. Der kleine Nachteil dieser Alternative ist der, daß die Aufrufe für die Standard E/A über das PBS10 als Zwischeninstanz erfolgen, wodurch eine im Verhältnis zur E/A-Gesamtdauer jedoch äußerst geringe Verzögerung entsteht.

3. Aufruf-Schnittstellen des PBS1Ø

3.1 Überblick

Einen Überblick über die Schnittstellen zum PBS1Ø gibt Abb.1 mit den Schnittstellen zur Anwenderebene (Ausführung von PEARL-Tasks) zur ORG-Ebene und zu den Treiberebenen. Das PBS selbst läuft auf einer eigenen Ebene (Registertafel) als Anwenderprogramm unter dem ORG ab.
Insgesamt belegen die folgenden Programmteile jeweils eine Ebene:

- PEARL-Prozesse
- PBS1Ø
- Sonstige, nicht PEARL-Programme
- Standard-Bedienprogramm
- ORG
- E/A-Treiber (Standard-E/A)
- Sonstige Treiber
- Uhrzeit-Behandlung
- Fehlerbehandlung

Im weiteren sind die Schnittstellen zwischen diesen Ebenen beschrieben, soweit sie das PEARL-Ablaufsystem betreffen.

3.2 Schnittstelle Laufzeitsystem-PBS1Ø

PEARL enthält (wie generell jede höhere Sprache) Operationen, die nicht unmittelbar auf den Befehlsvorrat des Zielrechners abbildbar sind und damit eine virtuelle Maschine als Schnittstelle voraussetzen. Der Befehlsvorrat dieser virtuellen Maschine ist durch das Laufzeitsystem realisiert.

Das PEARL-Laufzeitsystem für die S310 ist als Bibliothek organisiert und muß zu dem compilierten Anwenderprogrammsystem hinzugebunden werden. Alle Laufzeitroutinen sind reentrantfähig codiert und müssen deshalb in einem ablauffähigen PEARL-Programmsystem (mehrere Module, mehrere Tasks) nur einmal vorhanden sein.

Die Laufzeitroutinen lassen sich wie folgt klassifizieren:

- arithmetische Operationen, insbesondere Gleitpunktarithmetik
- Standardfunktion, z.B. ABS, MOD, LWB, SIN usw.
- Adressierung, insbesondere Adressberechnung für mehrdimensionale Felder
- Handhabung von Unterprogrammen
  - Parameterübernahme
  - dynamische Speicherplatzverwaltung für prozedur-lokale Daten, Reentrantfähigkeit
- formatierte und binäre E/A, insbesondere Verwaltung der geräte- und dateispezifischen Kontrollblöcke:
  - Datenaufbereitung
  - Pufferverwaltung, Synchronisation
  - Prüfung von Konsistenz und Zugriffsrechten
- Schnittstelle zur Prozeßverwaltung, d.h. Anbindung PEARL-Tasks an PBS1Ø

Die Laufzeitroutinen für Standard-E/A und Taskverwaltung rufen das PBS1Ø auf (vgl. Abb.1):

(1) Anforderungen an das PBS1Ø

- programmierte Anforderung (Maschinenbefehl)
- globaler Bereich zur Parameterübergabe

(2) Rückgabe der Kontrolle an PEARL-Anwenderprogramm

- programmierte Anforderung
- ggf. Anzeigenübergabe mit anschließender SIGNAL-Erzeugung

3.3 Schnittstelle ORG-PBS1Ø

Wie bereits erwähnt, werden ORG-Funktionen nur für die Standard-E/A über die Anwenderschnittstelle (R7 : US 12) vom PBS1Ø benutzt.
Leider besitzt das ORG bezügl. Wartefunktion keine einheitliche Schnittstelle:

(3) ORG-Aufruf durch PBS1Ø (Standard-E/A)

- mit implizitem Warten
- "Mehrfachwarten" mit Angabe eines Koordinierungszählers

④ Rückmeldungen des ORG

- PBS1Ø fortsetzen nach implizitem Warten
- Gerätefertigmeldung mit Koordinierungszähler freigeben

3.3.1 Aufrufe mit impliziten Warten

Bei ORG-Aufrufen mit Koordinierungszähler kann das aufrufende PBS1Ø und damit das PEARL-System simultan weiterlaufen. Die ORG-Aufrufe mit implizitem Warten setzen das PBS1Ø solange in einen Wartezustand bis der Aufruf vom ORG bearbeitet ist. Diese Aufrufe werden für einige E/A-Anforderungen verwendet, da dafür keine Aufrufe mit Koordinierungszählerangabe existieren. Dies gilt für das Einrichten, Löschen , Ändern der Dateilänge, Ändern des Dateinamens und Einlesen des Buchhalters einer Datei. Die Blockierung des PBS10 nach diesen E/A-Anforderungen hätte zwar durch Einführen einer Hilfsebene umgangen werden können; der zusätzliche Aufwand erschien jedoch nicht gerechtfertigt, da solche Aufrufe durch off-line Einrichten von Dateien auch vermeidbar sind.

3.3.2 Aufrufe mit Koordinierungszählern

Im ORG dient der Koordinierungszähler dazu, den Zugriff mehrerer Programme zu einem Betriebsmittel zu koordinieren. Im PBS10 wird für die E/A-Aufrufe mit Koordinierungszählerangabe die Funktion mit Mehrfachwarten benutzt. Diese ermöglicht es, mehrere E/A-Aufrufe an simultan arbeitsfähige Geräte abzugeben.

Dabei wird beim Belegen eines Betriebsmittels(Gerät) vorausgesetzt, daß alle darauf zugreifenden Prozesse von sich aus Koordinierungsmaßnahmen treffen; d.h. das E/A-Laufzeitsystem hat dafür zu sorgen, daß kein weiterer E/A-Aufruf an das ORG erfolgt, solange noch eine Rückmeldung des ORG für einen früheren Aufruf für dieses Gerät aussteht. Um diesen Mehrfachzugriff zum gleichen Gerät zu verhindern, wird der Zugriff zu einem E/A-Aufruf mit Koordinierungszähler über eine Semaphore synchronisiert, welche sich im Geräte-Kontrollblock des Gerätes befindet.

3.4 Schnittstelle Treiber-PBS1Ø

Für nicht Standard-Geräte und für Anschlüsse an das RDC-Kommunikationssystem (Netzbetriebssystem) stellt das PBS1Ø eigene Routinen (Treiber) zur Verfügung. Diese laufen auf den Anschlußstellen zugeordneten Ebenen ab, die Befehlszähler werden im PBS1Ø-Anlauf initialisiert. Die Verbindung zwischen Treibern und PBS1Ø erfordert besondere, hier nicht näher beschriebene Maßnahmen der Koordinierung, da diese nicht ohne weiteres den ORG-Koordinierungsmechanismen (programmierte Anforderung) überlappbar sind.

(5) Treiberanforderung an das PBS1Ø

- programmierte Anforderung
- Parameterübergabe in globalem Bereich

Für die Bearbeitung der Uhrzeit benötigt das PBS1Ø eine eigene Treiber-Routine. Diese wird vom PBS1Ø mit der Zeitdifferenz bis zur nächsten fälligen PEARL-Einplanung versorgt, nach Ablauf dieser Zeit erfolgt die Rückmeldung nach (5). Das ORG bietet die Möglichkeit der dynamischen Anbindung dieser PBS1Ø - Zeitroutine an die ORG-Standardzeitbearbeitung, bzw. den 20 ms Zeittakt der S310. Angabe der Adresse einer "Anwenderzeitgeberroutine" in < Hauptspeicherende - 10 >).

4. Grobstruktur des PBS1Ø

Die Abb.2 gibt ein vereinfachtes Zustandsdiagramm für PEARL-Zustände. Eine PEARL-Task kann sich demnach in den Zuständen bekannt, (lauffähig) laufend, wartend oder blockiert befinden. Zustandsänderungen werden eingeleitet durch Aufrufe zum Aktivieren, Suspendieren, Fortsetzen und Terminieren einer Task. Für Synchronisationsmechanismen stehen SEMA- und BOLT-Variablen zur Verfügung.

Dieses Zustandsdiagramm ist nicht vollständig. So enthält es z.B. keine Zeit- und Unterbrechungs-Einplanungen und es gibt keine Auskunft darüber, unter welchen Bedingungen die Übergänge stattfinden. Dennoch gibt es einen groben Eindruck der wichtigsten Zustände und ihren Übergängen, deren Verwaltung die wesentlichste Aufgabe des PBS1Ø ist.

Die Abb.3 gibt einen groben Überblick über die Struktur und die Funktionen des PBS1Ø.

Das PBS1Ø besteht im wesentlichen aus einem Initialisierungsteil, einem Verteiler (mit der sich anschließenden Behandlung der verschiedenen Aufrufe) und einem Dispatcher.

Nach der Initialisierung einiger Größen im Anlauf des PBS10 gibt dieses die Kontrolle an die höchstpriore lauffähige Task.

Das PBS1Ø kann danach wieder von einem Aufruf durch das Anwenderprogramm (PEARL-Aufruf) oder einer Unterbrechung (Treiber-Aufruf) oder einem Aufruf durch das ORG 310 (E/A-Rückmeldung vom ORG 310) aktiviert werden.

Nach einem PEARL-Aufruf folgt die Bearbeitung eines E/A-Auftrages oder einer Einplanung oder einer Synchronisierungsmaßnahme oder einer Operation, die eine PEARL-Zustandsänderung herbeiführt.

Nach einem Treiberaufruf erfolgt die Behandlung des Interrupts. Falls insbesondere die Unterbrechung als Folge der Erfüllung einer Zeiteinplanung verursacht wurde, erfolgt danach, wie bei

einem PEARL-Aufruf, eine Operation, die eine PEARL-Zustandsänderung herbeiführt.

Im Falle einer E/A-Rückmeldung vom ORG 310 erfolgt entweder die Rückmeldung mit Erhöhen des Koordinierungszählers oder Fortsetzen des wartenden PBS1Ø als Quittung auf einen ORG 310-Aufruf mit implizitem Warten.

Nach Bearbeitung der verschiedenen Anforderungen geht das PBS1Ø sodann in den Dispatcher, wo es die höchstpriore lauffähige Task sucht. Falls eine solche existiert, übergibt ihr das PBS1Ø dann den Prozessor. Falls eine solche nicht existiert, wird vom PBS1Ø die Kontrolle an das ORG übergeben.

Abb.4 zeigt eine Grobstruktur des Task-Kontrollblocks (TCB) einer Task. Er enthält die Task-spezifischen Daten zur Verwaltung eines Prozesses. Für jede Task wird vom Compiler ein solcher TCB reserviert. Dadurch ist es möglich, eine variable Anzahl von Tasks unter dem PBS1Ø ablaufen zu lassen.

Abb.5 zeigt die 4 Klassen von Warteschlangen, die vom PBS1Ø verwaltet werden. Es gibt eine Warteschlange, in der die Tasks nach ihrer Priorität geordnet stehen, Warteschlangen hinter Synchronisationsvariablen, Warteschlangen für Unterbrechungseinplanungen und eine für Uhrzeiteinplanungen. Die Warteschlangen sind über 2 Zellen im TCB (Vorgänger, Nachfolger) in sich doppelt verkettet. Der Anker für eine Warteschlange liegt in einem für sie vorgesehenen Kontrollblock.

5. Angaben über Speicherplatzbedarf und Laufzeiten

Speicherplatzbedarf:

Programmteile (Ass. codiert)	ungefährer Speicherplatzbedarf in KW
PBS1Ø + Uhrtakttreiber	2.8
Laufzeitroutinen (in Bibliotheken):	
- zur Verwaltung der Tasks	0.7
- für formatierte E/A in PEARL codiert	8.7
- für binäre E/A	3.2
- sonstige	4.2

Laufzeiten :

Aktion	ungefähre Laufzeiten in µs
PEARL Zustandsänderung (activate, suspend, continue, terminate)	550 - 600
PEARL Synchronisation (request, release)	550 - 700 (bei vorhandener Warteschlange) 65 (ohne Warteschlange)
Interrupt-Bearbeitung bis zur ersten PEARL-Anweisung	600
Bearbeitung des Standard E/A-Aufrufes im PBS1Ø (Aufruf + Rückmeldung)	400

Die Laufzeiten ergeben sich aus der Anzahl der Assembler-Anweisungen und einer mittleren Zeit zum Ausführen eines Befehls.

6. Benutzerhinweis

Für die Benutzung des PBS1Ø ergeben sich verschiedene Alternativen aus den Möglichkeiten, die das ORG beim Urladen und denen, die eine Anwenderanlaufroutine bietet. Zum Beispiel kann das PBS1Ø beim Urladen des ORG mit eingelagert werden. Durch eine Anwenderanlaufroutine kann der Anwendermodul eingelagert und das PBS1Ø gestartet werden (nach dem Anlauf des PBS1Ø gibt dieses dann die Kontrolle an das Anwenderprogramm).

7. Ausblick: Ein PEARL-Testsystem

Zur Zeit befinden wir uns in der Konzeptphase für ein PEARL-Testsystem, d.h. Ansprache von Testobjekten mit den Bezeichnern der PEARL-Quelle. Dieses wird in PEARL programmiert werden und etwa den folgenden Funktionsumfang haben:

- Operationen bezügl. Haltepunkten (Angabe als PEARL-Quellzeile)
- Auskunft und Veränderungsmöglichkeiten für Variablen, Tasks, Synchronisationsvariablen
- Möglichkeiten, Unterbrechungen zu erzeugen
- Möglichkeiten für eine Analyse nach Abbruch eines Programms (post mortem).

Dafür werden vom Compiler 2 Listen abgesetzt. Die erste Liste enthält eine Zuordnung:

Programmzeile - Adresse (Offset zum Modulbeginn)

Die zweite Liste enthält eine Zuordnung:

Name - Adresse $\left\langle \begin{matrix} \text{global} \\ \text{lokal} \end{matrix} \right\rangle$ Datentyp

Darüber hinaus muß das PBS1Ø entsprechende Testsystemanschlüsse bereitstellen.

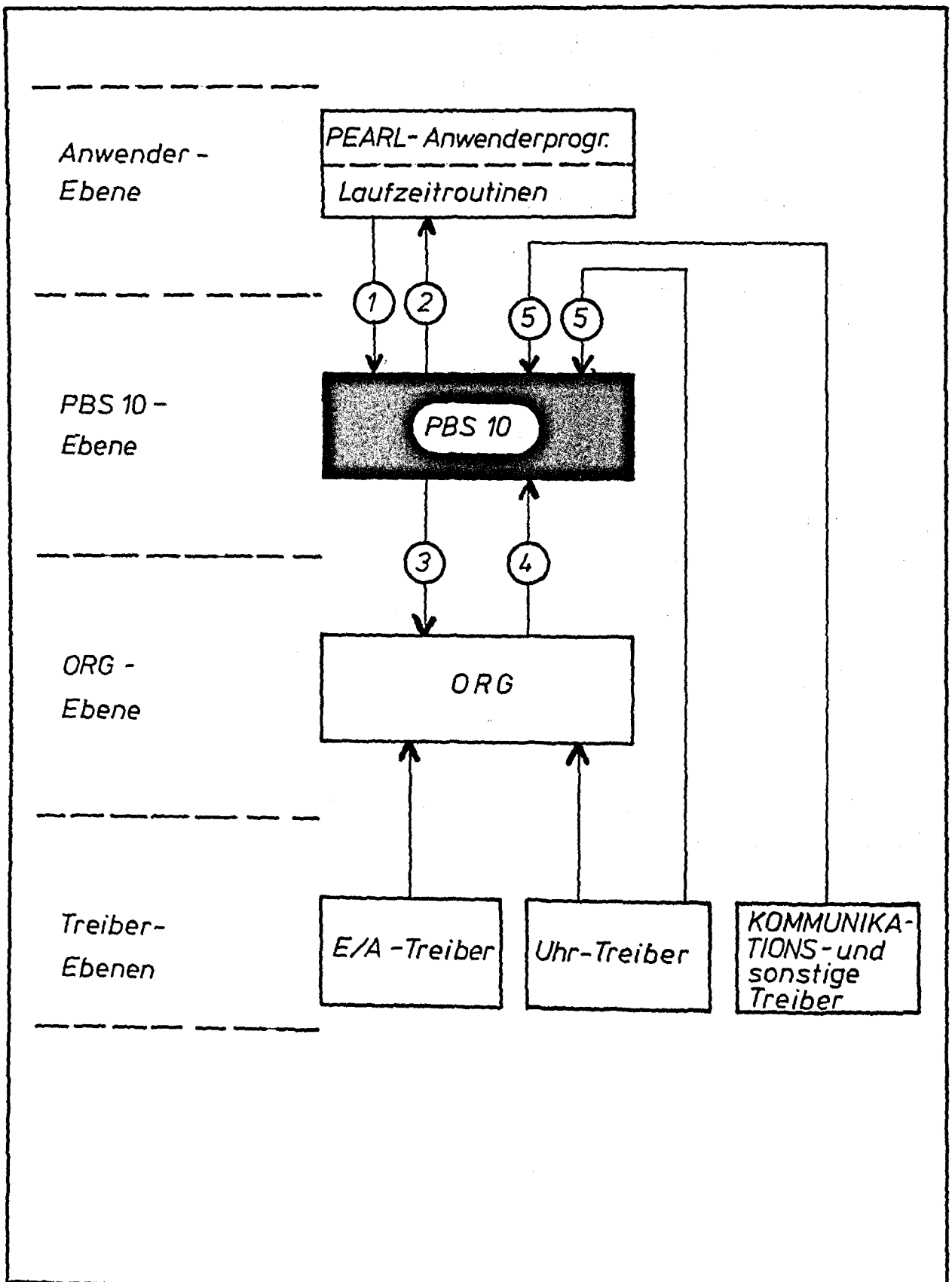


Abb. 1
Schnittstellen zum PBS 10

Fraunhofer-Institut für Informations- und Datenverarbeitung · IITB

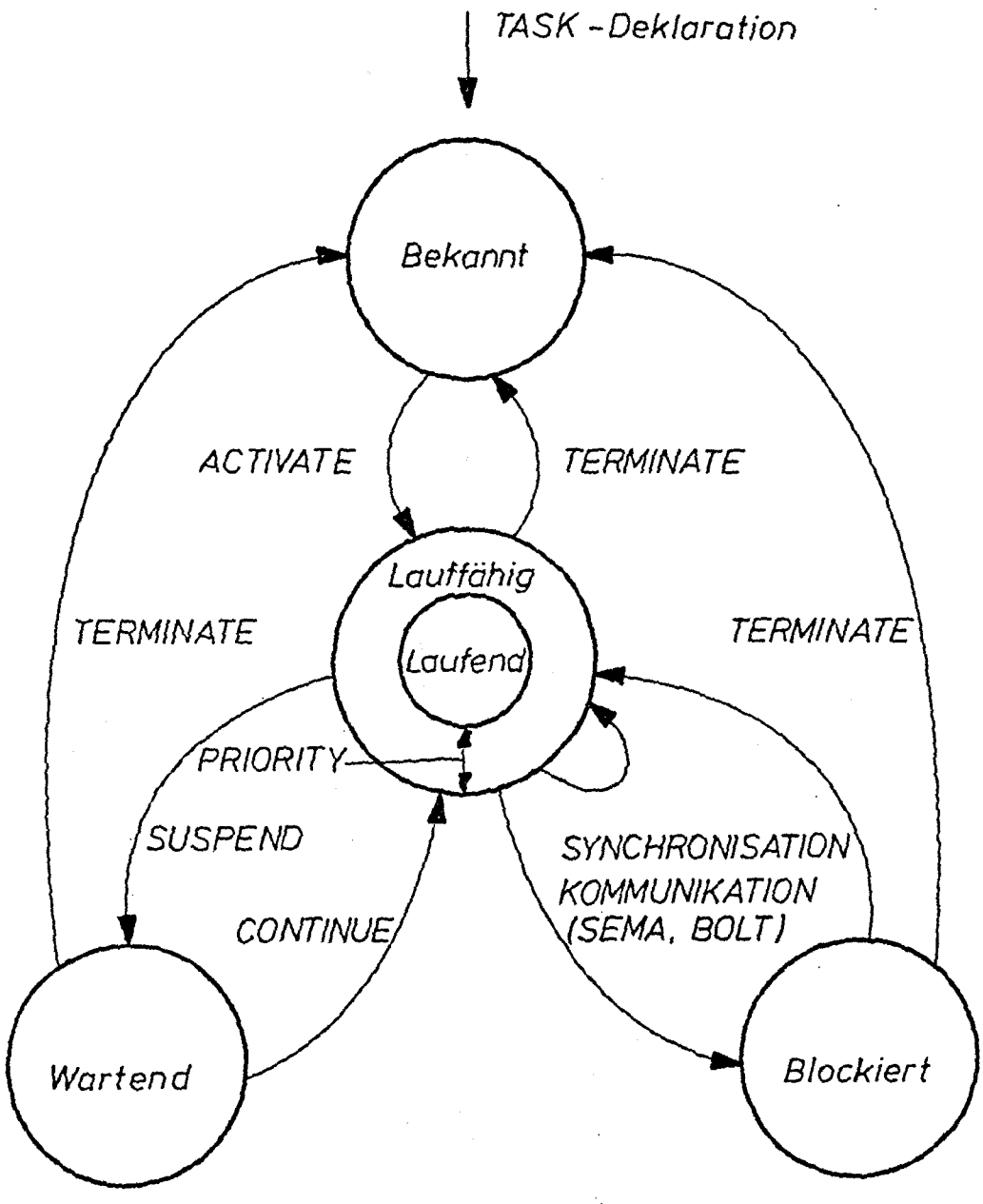


Abb. 2 Zustandsdiagramm für PEARL - Tasks

Fraunhofer-Institut
für Informations- und
Datenverarbeitung · IITB

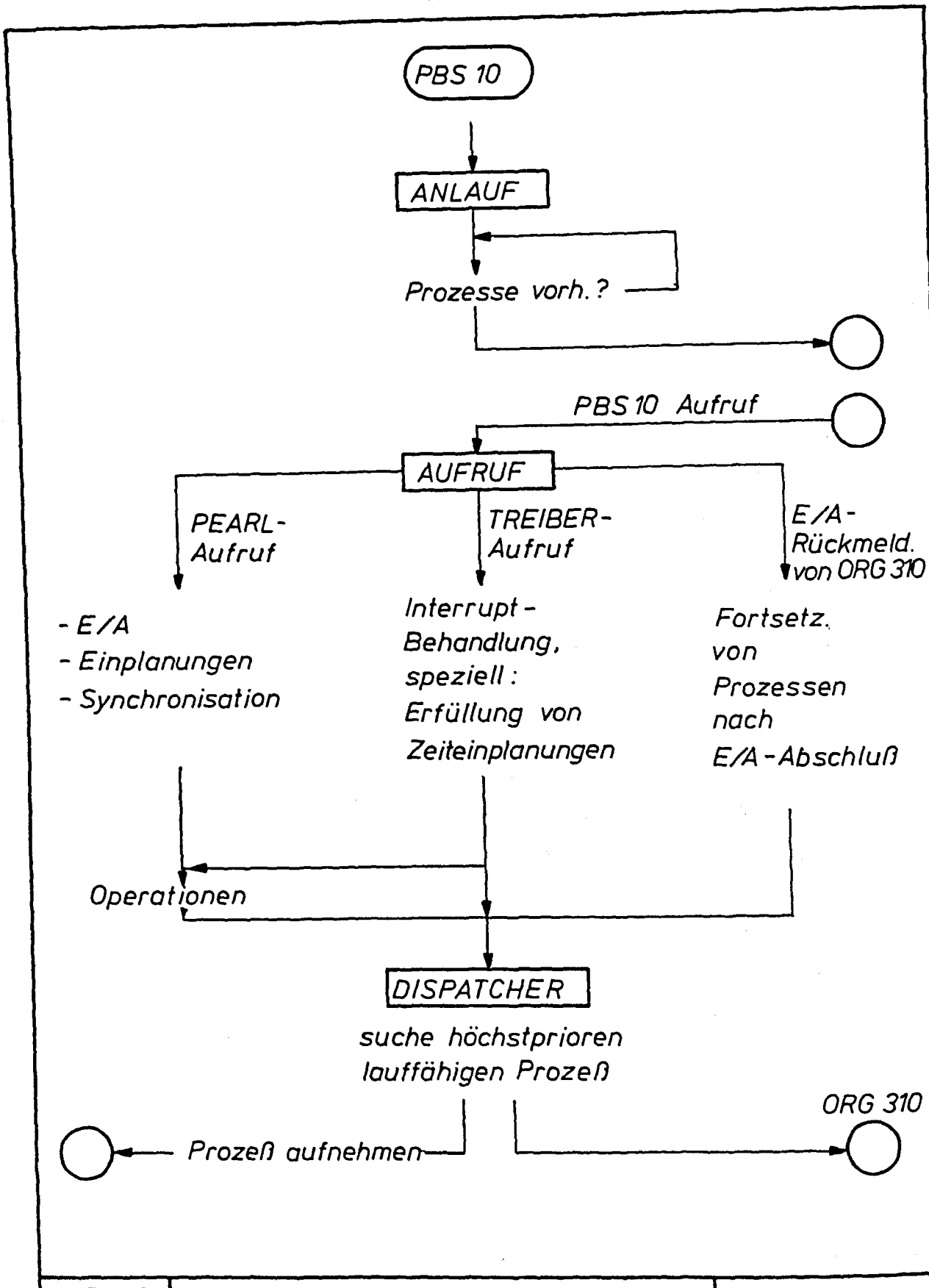


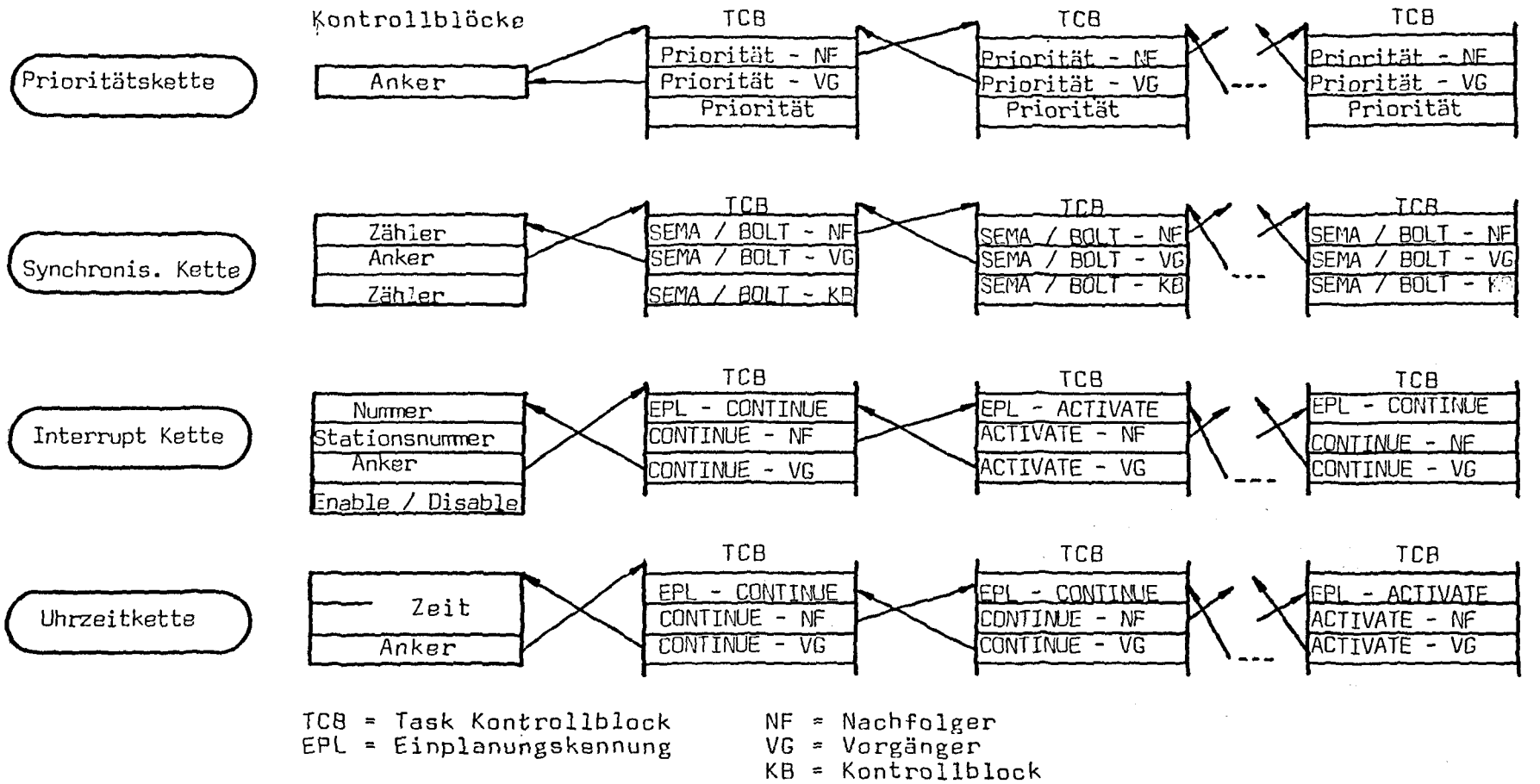
Abb. 3
Grobstruktur des PBS 10

Fraunhofer-Institut
für Informations- und
Datenverarbeitung · IITB

Der TCB enthält in 53 Worten:

- Task Zustand
- Task Priorität
- Task Startadresse
- Task Name
- Verkettung der Prioritäts-Warteschlange
- Verkettung der SEMA-, BOLT-Warteschlangen
- Verkettung der Einplanungs-Warteschlangen
- Registersatz

Abb. 4: Inhaltsverzeichnis des Task Kontrollblocks (TCB)



TCB = Task Kontrollblock
EPL = Einplanungskennung

NF = Nachfolger
VG = Vorgänger
KB = Kontrollblock

Abb. 5

PEARL-Betriebssystem Datenstrukturen und Verzeigerungen (Warteschlangen)

Fraunhofer-Institut für Informations- und Datenverarbeitung · IITB

I. Hertlin, L. Lorenz

Fraunhofer - Institut
für Informations- und Datenverarbeitung
IITB
Karlsruhe

PEARL-Programmierung auf der SIEMENS 310

Übersetzungssystem

Zusammenfassung:

Ein vom Entwicklungsbüro Wulf WERUM, Lüneburg, entwickelter portabler PEARL-Übersetzer wurde im IITB auf einer Siemens 310 installiert und erzeugt Programme für diese Maschine. Der Übersetzer verarbeitet einen Sprachumfang, der erheblich über Basis-PEARL (DIN 66 253) hinausgeht. Der Sprachumfang umfaßt insbesondere Referenzen, Typ- und Operatorvereinbarungen. Ein im Übersetzer enthaltener Preprozessor ermöglicht bedingtes Übersetzen und Einkopieren von Quellsprache aus weiteren Dateien oder Bibliothekselementen. Der Übersetzer ist in GBL1, einem PL/1-Subset, geschrieben. Ein maschinenunabhängiger Übersetzer-Oberteil übersetzt die Quellsprache in eine maschinenunabhängige Zwischensprache IL1. Ein maschinenspezifischer Codegenerator erzeugt daraus im vorliegenden Fall einen Subset der Assemblersprache ASS300, der vom Übersetzer AS10 verarbeitet werden kann. Da der GBL1-Übersetzer von WERUM ebenfalls IL1 erzeugt, konnte der PEARL-Übersetzer mit Hilfe des 310-Codegenerators selbst auf die 310 gebracht werden. Für Segmentierung und Datenpaging mußten Programmpakete entwickelt werden, da die Systemprogramme der 310 hierfür keine Unterstützung bieten. Um Betrieb, Installation und Wartung des Übersetzers zu erleichtern, wurde ein Monitor und ein komfortables Dienstprogramm zur Pflege von Datenbeständen auf Peripherspeichern entwickelt.

Über die aus der Sicht des Anwenders wichtigen Eigenschaften und Leistungsdaten des Übersetzers und geplante Weiterentwicklungen wird berichtet.

Inhalt :

1. Entwicklungsziele

Das Fraunhofer-Institut für Informations- und Datenverarbeitung (IITB) hat im Rahmen des Forschungsprojektes "Echtzeitrechnersystem mit verteilten Mikroprozessoren" /1/ ein PEARL-Programmiersystem hierfür erstellt, das auf einem Prozeßrechner 310 K läuft und die Programmierung sowohl der Mikrorechner als auch der 310 in PEARL erlaubt.

Der vorliegende Beitrag beschreibt Aufbau und Eigenschaften des Übersetzungssystems. Mit dem zugehörigen Betriebs- und Laufzeitsystem befaßt sich Heine /2/; über Erfahrungen mit dem Gesamt-Programmiersystem berichtet Kippe /3/ (beide in diesem Band).

2. Systemumgebung

Den gerätetechnischen Aufbau des eingangs erwähnten Echtzeitrechnersystems, das unter der Kurzbezeichnung RDC-System (Really Distributed Control) bekannt geworden ist, zeigt Bild 1.

Es besteht aus einer Anzahl auf Mikroprozessorbasis (SAB 3000) aufgebauten Rechnerstationen, die in räumlich weit ausgedehnten Industrieanlagen möglichst nahe am Ort des zu steuernden Prozesses installiert werden und über eine ringförmige Sammelleitung miteinander und einem Programmier- und einem Prozeß-Leitrechner verbunden sind. Bei den beiden letzteren handelt es sich um Prozeßrechner Siemens 310 K. Sie haben über einen Ein-/Ausgabe-Umschalter Zugriff auf eine Plattenspeicher-Steuerung mit mehreren angeschlossenen Laufwerken. Zur Prozeßführung wird ein Ein-/Ausgabe-Farbbildschirmsystem mit ggf. mehreren Bedienplätzen eingesetzt, das vom IITB im Rahmen eines anderen Forschungsvorhabens /4/ entwickelt wurde und inzwischen von Siemens in Lizenz gebaut und vertrieben wird.

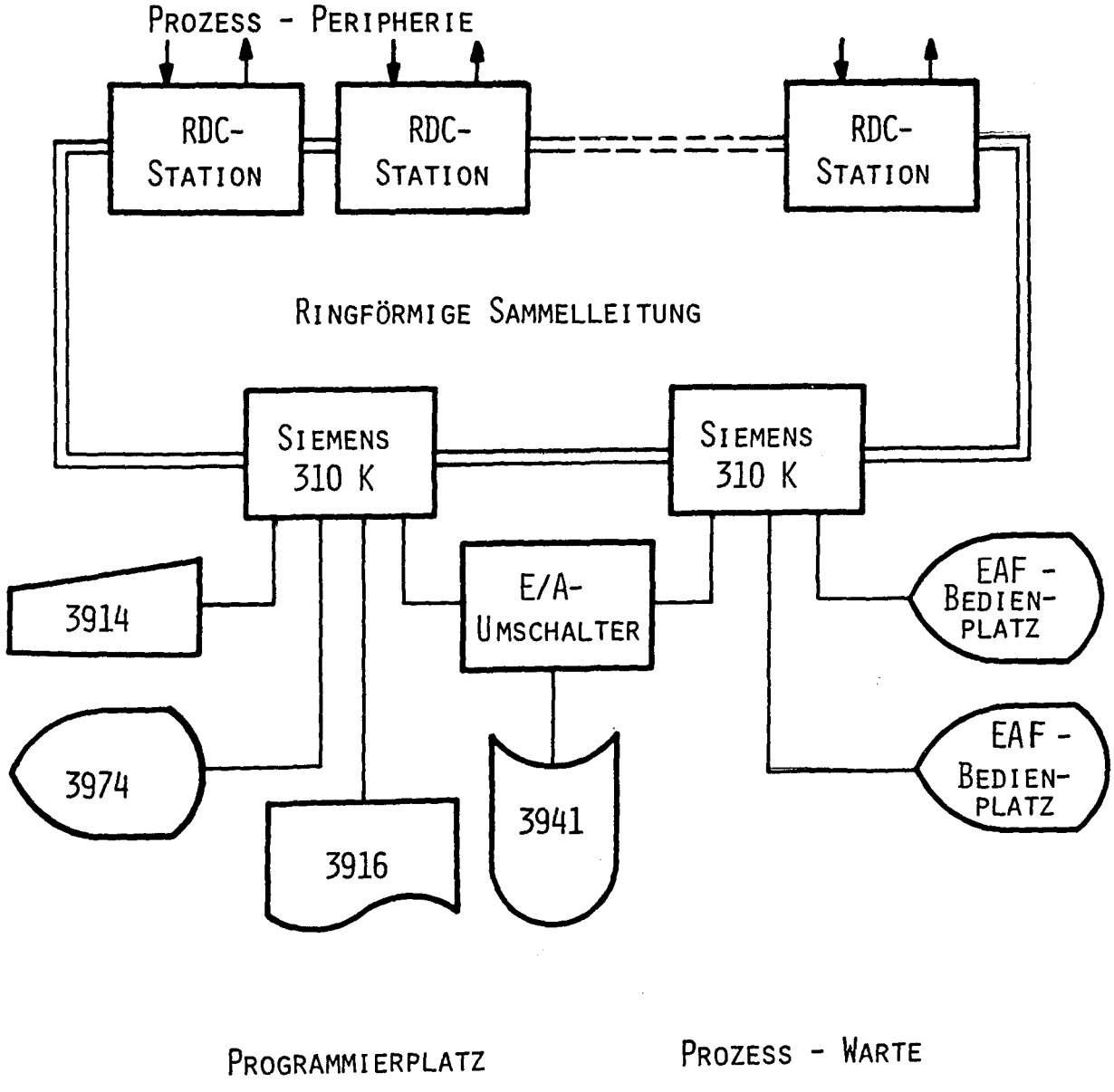


BILD 1: KONFIGURATION DES RDC-SYSTEMS

Fraunhofer-Institut
für Informations- und
Datenverarbeitung · IITB

Die RDC-Stationen besitzen nur Prozeßperipherie. Sie werden vom Programmierrechner über die Sammelleitung mit Programmen geladen und gestartet. Die Entwicklung der Anwenderprogramme erfolgt ausschließlich in PEARL.

3. Aufbau des Programmiersystems

Den Gesamtaufbau des Programmiersystems zeigt Bild 2. Der PEARL-Übersetzer kann aufgrund der im Quellmodul enthaltenen %INCLUDE-Anweisungen noch weitere Quellspracheteile aus Dateien einkopieren.

Er übersetzt in einen Subset der Assemblersprache ASS300, der von dem auf 310 lauffähigen Assembler AS10 verarbeitet werden kann. Die von AS10 erzeugte Grundsprache wird ggf. mit weiteren aus PEARL erzeugten Grundsprache-Moduln und mit einer oder mehreren Systembibliotheken gebunden. Der Ladebinder LB10 legt das Gebinde als absolutierten Maschinencode im Hauptspeicher der 310 ab. Ein Programm PSIO gestattet, das gebundene Programm auf Platte oder Diskette aus- und ggf. in den gleichen Hauptspeicherbereich wieder einzulagern. Es kann aber auch mit dem RDC-Lader LAD1 in eine RDC-Station geladen werden.

In einer weiteren, noch in Arbeit befindlichen Ausbaustufe des Systems erzeugt der Übersetzer direkt Grundsprache mit PEARL-spezifischer Zusatzinformation, z.B. zur Verträglichkeitsprüfung Globaler Objekte. Ein PEARL-Binder PB10 erzeugt als Ergebnis des Bindeprozesses wieder verschiebliche Grundsprache, die mit dem Lader des ORG310 in die 310 oder mit dem RDC-Lader LAD2 in RDC-Stationen geladen werden kann.

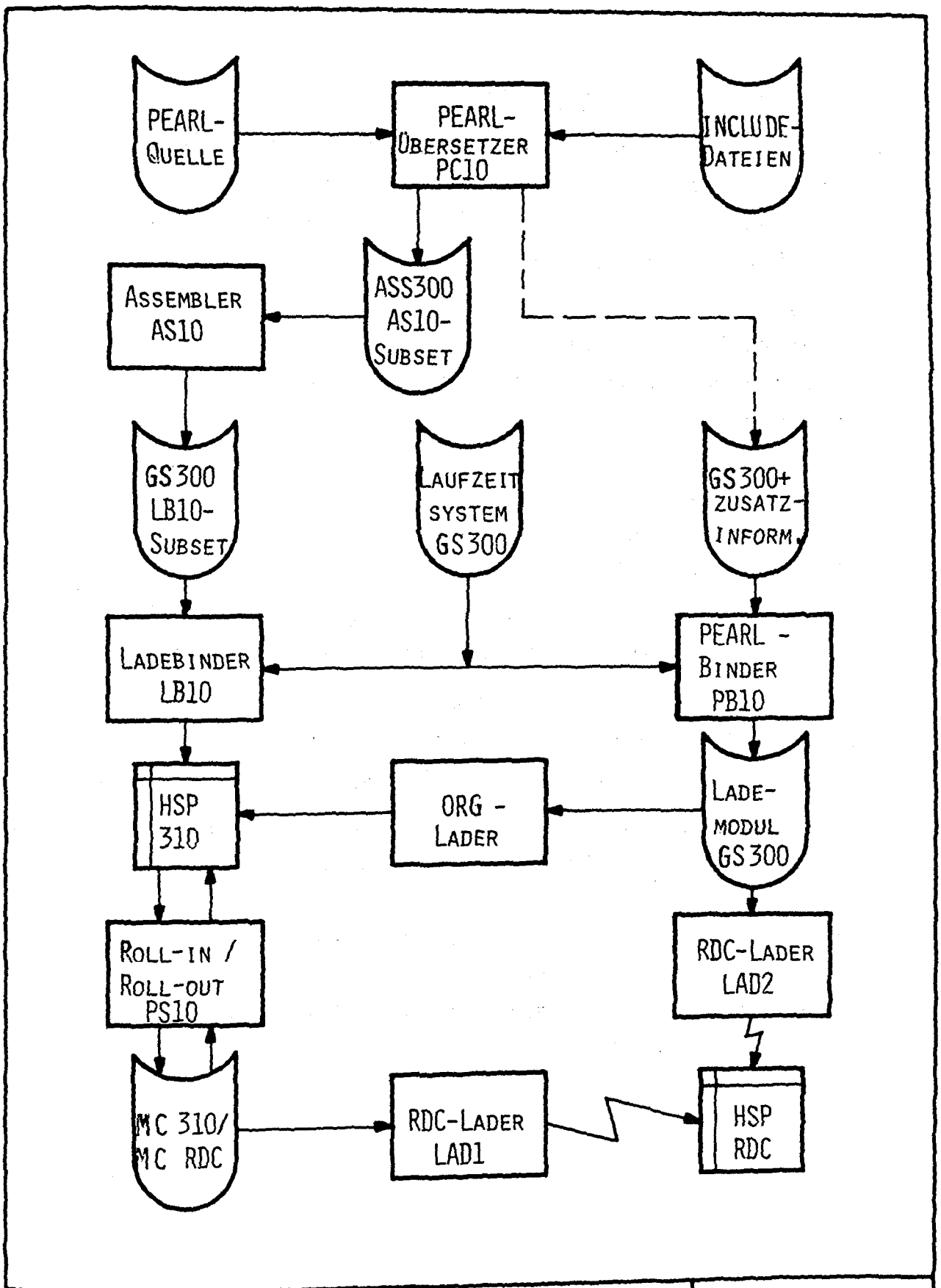


BILD 2: DATENFLUSS IM PROGRAMMIER-SYSTEM

Fraunhofer-Institut
für Informations- und
Datenverarbeitung · IITB

4. Implementierter Sprachumfang

Der Umfang des zugrundegelegten PEARL-Subsets ist ausführlich in /5/ beschrieben. Er umfaßt vollständig Basis-PEARL nach DIN 66253, Teil 1 /6/, darüber hinaus im wesentlichen folgende Sprachkonstrukte von Full PEARL /7/:

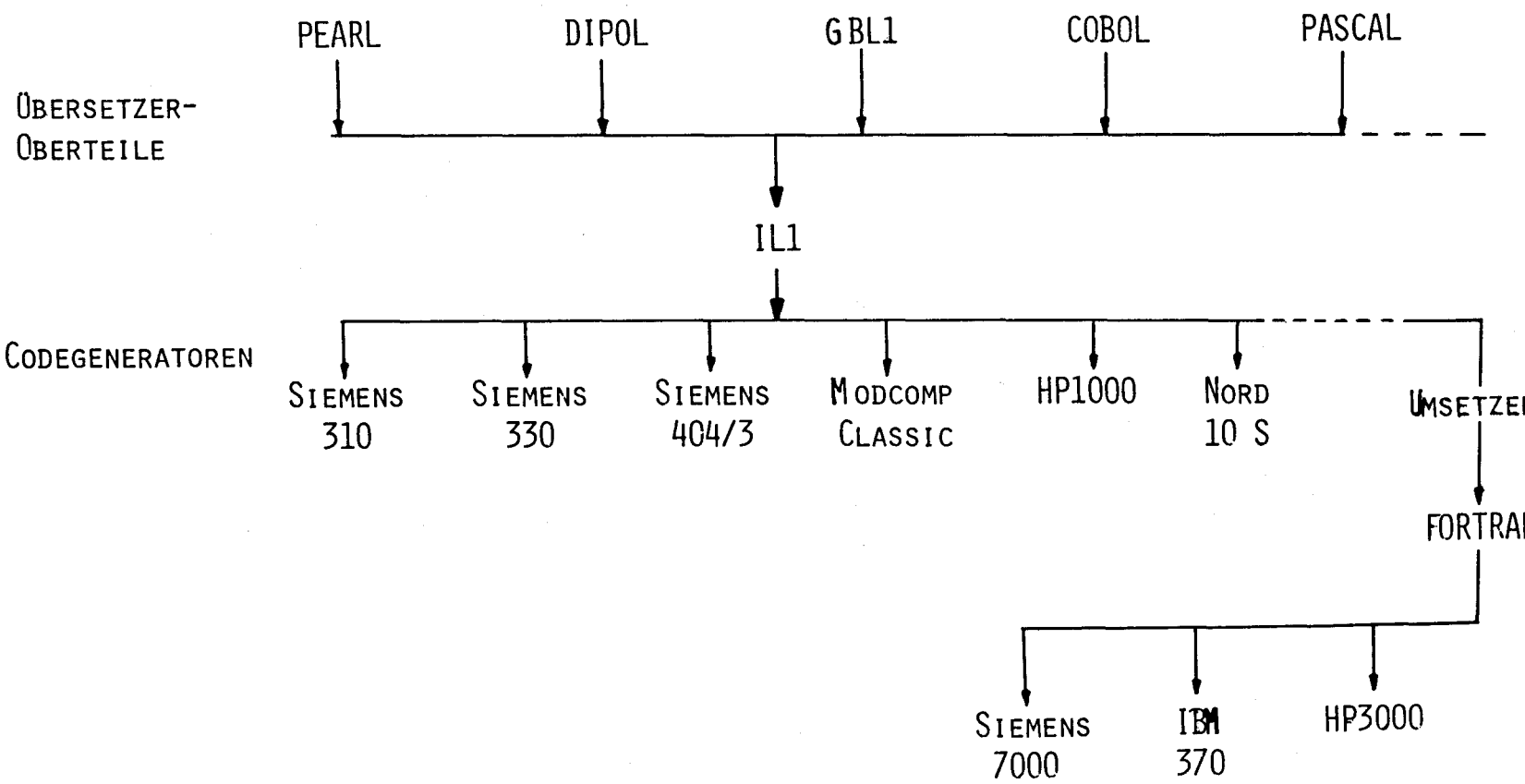
- Referenzen,
- mehrstufige Strukturen,
- Vereinbarung neuer Typen und Operatoren,
- BOLT-Synchronisation.

Zur Programmierung von Mehrrechner-Systemen werden außerdem nach einem Vorschlag von Steusloff /8/ folgende Spracherweiterungen aufgenommen:

- ein "Stationsteil" zur Beschreibung unterschiedlicher Rechnerstationen in heterogenen Mehrrechnersystemen,
- ein "Ladeteil" zur Beschreibung der Verteilung von Programm-Modulen auf die Rechnerstationen und zur Formulierung von Rekonfigurationsstrategien bei Teilausfällen,
- Erweiterungen des Systemteils zur Beschreibung alternativer Datenwege und zur Ersatzwertbeschaffung.

Ein im Übersetzer enthaltener Preprozessor erlaubt

- bedingtes Übersetzen und
- Einkopieren von Quellsprache-Abschnitten aus getrennten Quellsprache-Dateien.



werum

BILD 3: FAMILIE PORTABLER ÜBERSETZER

ENTWICKLUNGSBÜRO WULF WERUM
– DATENVERARBEITUNGSSYSTEME –

5. Übersetzer

Der Übersetzer gehört zu einer Familie portabler Übersetzer, die das Entwicklungsbüro Wulf WERUM, Lüneburg, entwickelt hat und die in Bild 3 dargestellt ist.

5.1 Aufbau

Der Übersetzer gliedert sich in einen weitgehend rechnerunabhängigen Oberteil, der einen PEARL-Modul in die rechnerunabhängige Zwischensprache IL1 übersetzt, und einen rechnerspezifischen Codegenerator, der daraus den Code für den jeweiligen Zielrechner erzeugt.

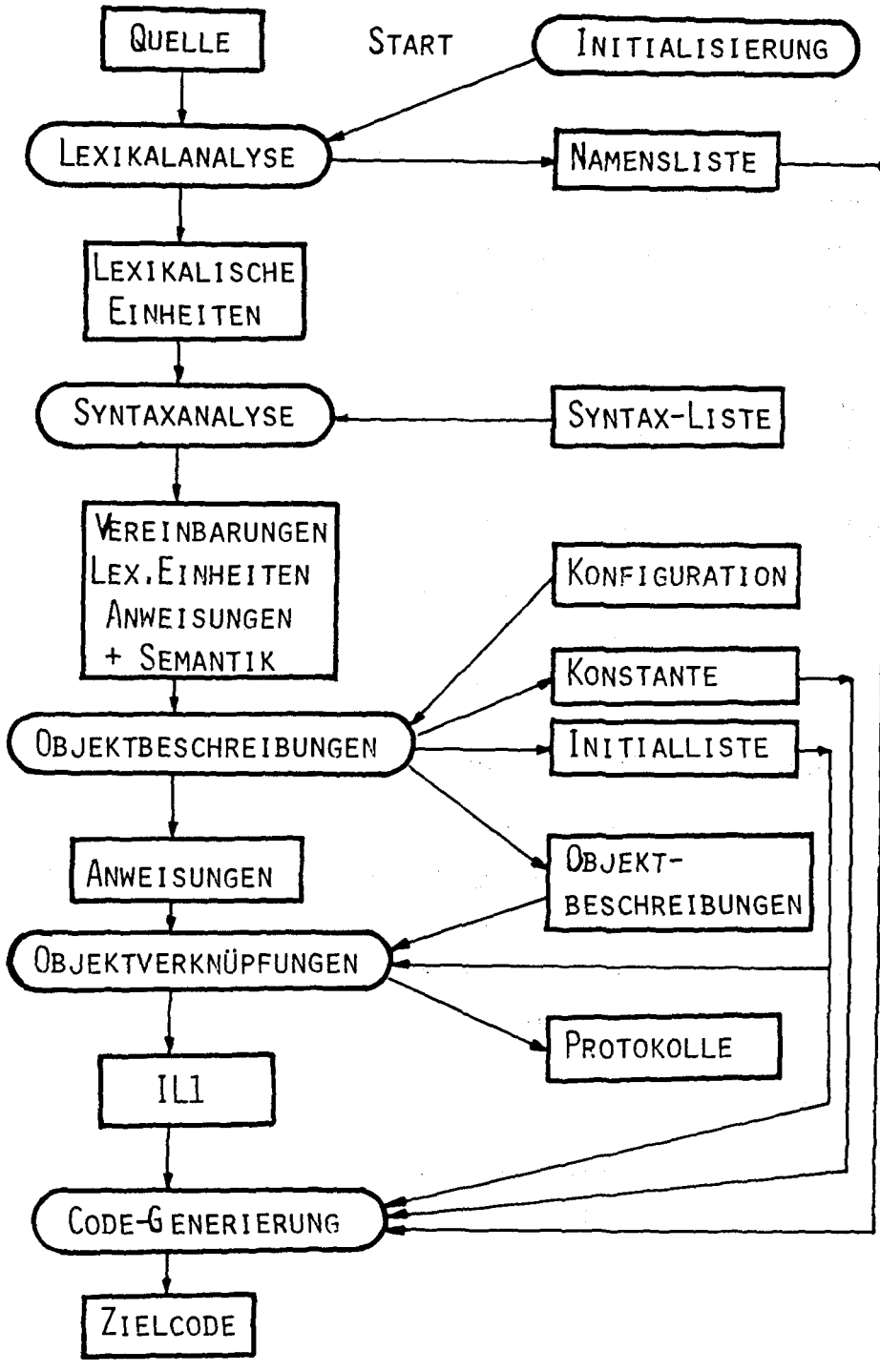
Den Übersetzer- Oberteil für Standard-PEARL hat das IITB von WERUM bezogen. Erweiterungen für Mehrrechner-PEARL und Codegenerator wurden im Auftrag und nach Spezifikationen des IITB ebenfalls von WERUM erstellt.

Stations- und Ladeteilübersetzer mit dem zugehörigen dynamischen Lader sind ebenso wie das Laufzeit- und Betriebssystem Eigenentwicklungen des IITB.

Da die mikroprogrammierte Struktur und Befehlsliste der RDC-Rechner eng mit der des Systems 300 verwandt ist, war es möglich, mit einem Codegenerator auszukommen, der nur den gemeinsamen Befehlsvorrat benutzt, und die Unterschiede durch angepaßte Betriebs- und Laufzeitsysteme auszugleichen.

Die Grobstruktur des Übersetzers zeigt Bild 4.

Damit er auch auf kleinen Rechenanlagen ablaufen kann, sind Oberteil und Codegenerator weiter in insgesamt 15 Phasen gegliedert. Beim Ablauf des Übersetzers befindet sich jeweils nur eine Phase im Arbeitsspeicher; der Übersetzer organisiert selbst die dynamische Überlagerung seiner Phasen in dem ihm zugeordneten Laufbereich des Arbeitsspeichers.



werum

BILD 4: GROBSTRUKTUR DES PEARL-ÜBERSETZERS

ENTWICKLUNGSBÜRO WULF WERUM
– DATENVERARBEITUNGSSYSTEME –

Bei der folgenden Phasen-Tabelle steht PL für PEARL:

- Initialisierung

  (0) INIMIX

- Analytischer Teil

  (1) PLLX1 Lexikalische Analyse 1
  (2) PLLX2 Lexikalische Analyse 2
  (3) PLSORT Sortierung der Bezeichner
  (4) PLBZUM Bezeichner-Umbenennung
  (5) PLSY Syntaxanalyse

- Auswertung der Objektbeschreibungen

  (6) PLSYST Auswertung des Systemteils
  (7) PLVM Aufbau des Vormerkbuches
  (8) PLAD Adressierung

- Auswertung der Objektverknüpfungen

  (9) PL12 Erzeugung von IL2
  (10) PLKVBZ Erzeugung der Kurz-Vormerkbücher
  (11) PL11 Erzeugung von IL1
  (12) PLFD Fehlermeldungen drucken
  (13) PLCR Crossreferenz-Liste drucken

- Code-Generierung

  (14) PLCD1 Codegenerierung 1
  (15) PLCD2 Codegenerierung 2

5.2 Die Zwischensprache IL1

IL1 ist eine compiler-interne Zwischensprache, die weitgehend rechner- und quellsprachenunabhängig ist. Sie eignet sich als Eingabe für die Code-Generierung in Übersetzern ebenso wie für Programm-Analysen.

Sie beschreibt die Verknüpfungen der im Programm vereinbarten Objekte in folgender Form:

Eine IL1-Operation besteht aus einem n-stelligen Operator in Präfix-Notation und n Operanden-Identifikationen.

Letztere können sein

- Hinweise auf eine Objektbeschreibung in einem zur Sprache gehörigen "Vormerkbuch",
- Adressen, wenn die Adreßvergabe für das Objekt schon erfolgt ist,
- Hinweise auf vorhergehende IL1-Operationen, deren Ergebnis der Operand ist.

Organisatorische Operationen erlauben es, die Anweisungs- und Blockstruktur des Quellprogrammes darzustellen. Compilerintern wird IL1 aus Geschwindigkeitsgründen als Folge von BIT(16)-Größen dargestellt. Zu Test- und Dokumentationszwecken ist eine lesbare externe Darstellung ähnlich einer Assemblersprache verfügbar.

5.3 Portabilität

Der PEARL-Übersetzer ist fast vollständig in GBL1, einem PL/1-Subset geschrieben. Der in Bild 3 erwähnte GBL1-Übersetzer wandelt ihn in IL1 um. Mittels des Codegenerators für die 310 war es daher möglich, den Übersetzer selbst auf diesen Rechner zu bringen.

Dazu waren in Assemblersprache zu codieren:

- vom Übersetzer benötigte Teile des PEARL-Laufzeitsystems,
- Routinen zum Ansprechen von
  - . Bediengerät
  - . Quellsprache-Eingabe
  - . Zielsprache-Ausgabe
  - . Protokoll-Ausgabe und
  - . Arbeitsdatei des Übersetzers auf Massenspeicher.

Letztere umfassen lediglich die Abbildung der Operationen

- Puffer lesen/schreiben
  (bei Plattenspeicher Pufferlänge = Blocklänge) und
- Datei einrichten /öffnen /schließen

auf die entsprechenden Aufrufe des ORG310.

Den Ablauf der sich im Hauptspeicher überlagernden Übersetzerphasen organisiert der Übersetzer selbst. Das Binden der Phasen in sich und mit dem Rootsegment kann mit dem Ladebinder LB10 erfolgen. Lediglich zum Austransfer der gebundenen Phasen in die Arbeitsformdatei mußte ein Programm geschrieben werden, das die Dateiverwaltung im Rootsegment des Übersetzers mitbenutzt. Diese stützt sich ebenso wie die Eingabe des Quell- und die Ausgabe des Zielprogrammes auf die Dateiorganisation des ORG310 ab. Eine Bibliotheksorganisation wie bei den größeren Rechnern des Systems 300 existiert auf der 310 nicht.

Einen Überblick über den gesamten Installationsweg gibt Bild 5.

5.4 Struktur und Eigenschaften des erzeugten Codes

5.4.1 Speicheraufteilung und Adressierung

Aus einem PEARL-Modul, der mehrere parallel ablauffähige Tasks enthalten kann, entsteht ein Assembler- und schließlich ein Grundsprache-Modul. Durch Binden der zusammengehörigen, aus PEARL erzeugten Grundsprache-Moduln mit dem

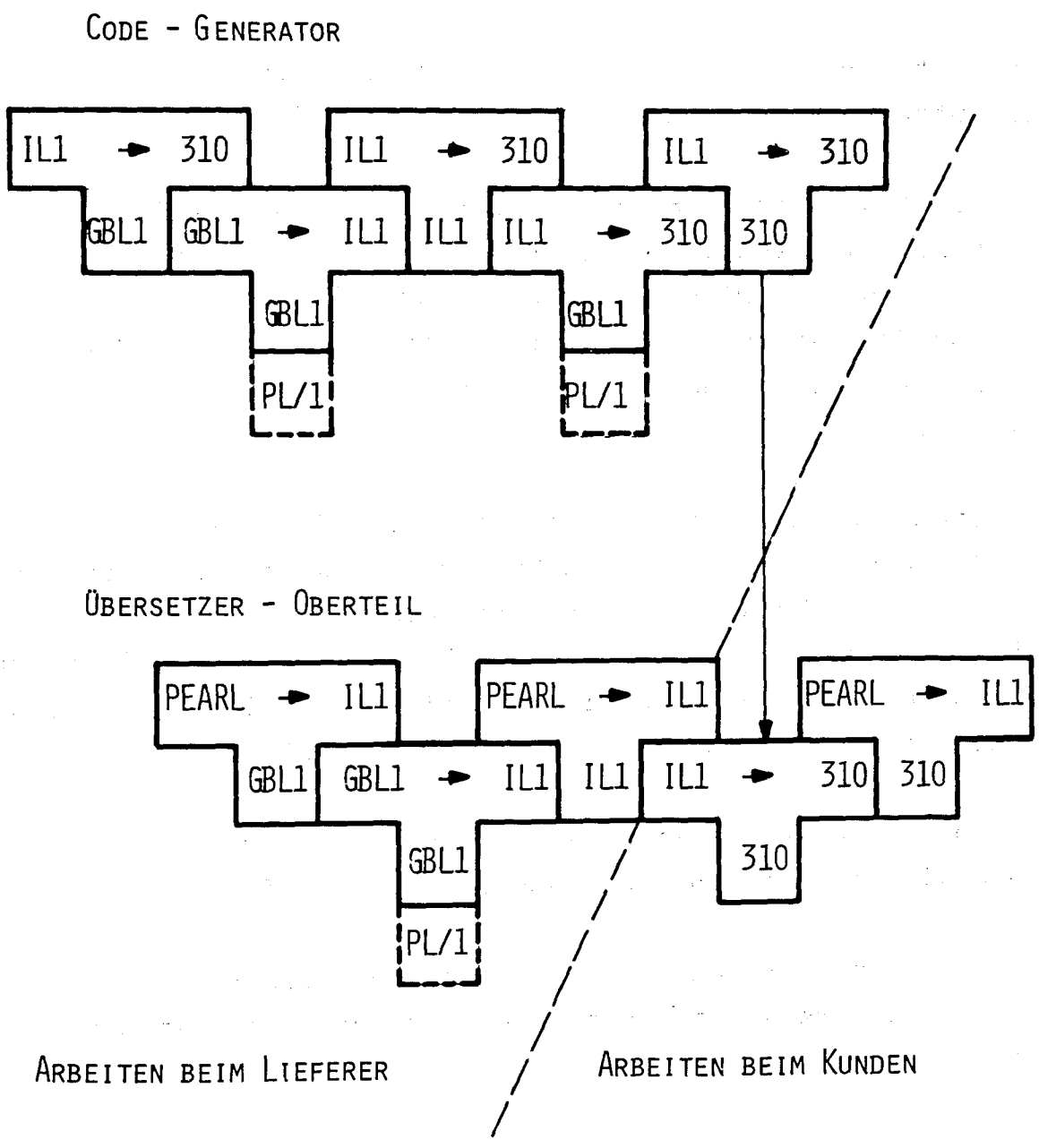


BILD 5: "BOOTSTRAPPEN" DES ÜBERSETZERS

Fraunhofer-Institut
für Informations- und
Datenverarbeitung · IITB

Betriebs- und Laufzeitsystem entsteht schließlich das ablauffähige Programm.

Die Gesamtanordnung im Hauptspeicher zeigt Bild 6 a, die innere Struktur eines übersetzten PEARL-Moduls Bild 6 b.

Beim Anlauf ruft das PEARL-Betriebssystem in jedem PEARL-Modul eine Anlaufroutine auf, welche die im Modul vorhandenen Tasks dem Betriebssystem bekanntmacht. Danach wird eine besonders gekennzeichnete Starttask aktiviert. Der weitere Ablauf erfolgt gemäß den PEARL-Anweisungen zur Tasksteuerung.

Die Systemprogramme der 310, wie Assembler, Binder und Lader ermöglichen keine Trennung von Programmen im varianten und invarianten Teil. Daher nimmt auch der PEARL-Übersetzer keine solche Trennung vor. Er trennt innerhalb eines Moduls Code und Daten, ordnet letztere aber in der Reihenfolge ihrer Deklaration an ohne sie nach variant und invariant zu sortieren.

Tasks können in dem verwendeten PEARL-Subset nur auf Modulebene deklariert werden, Subtasks sind nicht möglich.

Auf Modulebene oder innerhalb von Tasks deklarierte Daten werden statisch adressiert, wobei sich die Daten innerer Blöcke überlagern.

5.4.2 Befehlsvorrat

Im Befehlsvorrat der 310 nicht vorhandene Operationen wie Multiplikation, Division und die gesamte Gleitpunktarithmetik werden über Unterprogramme des Laufzeitsystems realisiert. Die im ORG310 verfügbaren Simulationsroutinen werden aus Gründen der Geschwindigkeit und des Unterbrechungsverhaltens nicht verwendet.

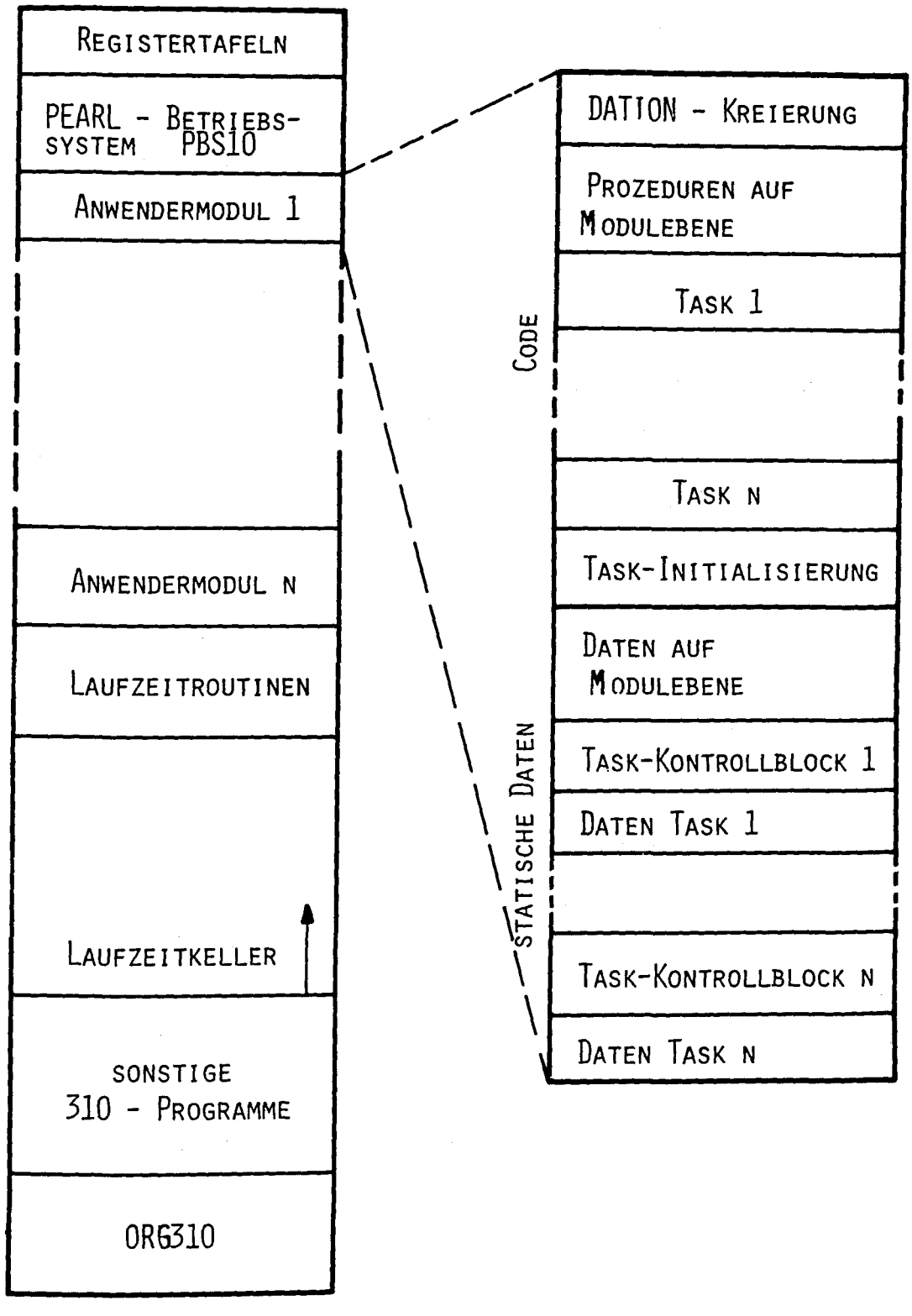


BILD 6: BELEGUNG DES HAUPTSPEICHERS

Fraunhofer-Institut
für Informations- und
Datenverarbeitung · IITB

5.4.3 Eintrittsinvarianz von Prozeduren

Auf Modulebene deklarierte Prozeduren können von parallelen Tasks "gleichzeitig" benutzt werden. Sie werden deshalb grundsätzlich "reentrant" codiert. Dies erfordert für jede parallele Aktivierung einen eigenen Bereich für die lokalen Daten der Prozedur, eine sogenannte "lokale Schachtel". Diese Bereiche werden in einem Laufzeitkeller vom Ende des freien Speichers absteigend dynamisch angelegt. Die Adressierung innerhalb einer "lokalen Schachtel" erfolgt relativ zu dem als Basisregister verwendeten R2. Rekursiver Aufruf solcher Prozeduren ist zulässig, wobei die Größe des Laufzeitkellers die mögliche Rekursionstiefe begrenzt.

Die lokalen Daten innerer Prozeduren werden in der lokalen Schachtel ihres statischen Vorgängers fest allokiert, um den Zeitbedarf für die dynamische Allokierung einzusparen und die Parameterübernahme zu vereinfachen. Dadurch ist für innere Prozeduren rekursiver Aufruf nicht zulässig.

5.4.4 Parameterübernahme

Am Anfang einer Prozedur steht für jeden Parameter ein Aufruf einer Parameterübernahmeroutine, die stets die Adresse des Parameters in einem Register liefert. Die Unterscheidung nach INIT-(Wertaufruf) oder IDENT-Parameter (Referenzaufruf) erfolgt im weiteren Prozedurcode.

Im Prozeduraufruf steht für jeden Aktualparameter eine Beschreibung, bestehend aus

- einem Verweis auf eine Zelle, die die Basisadresse der Schachtel enthält, in der der Parameter allokiert ist, und
- einer Distanzadresse zum Anfang dieser Schachtel.

Beim Aufruf innerer Prozeduren entfällt der erste Term, wenn der Aktualparameter ebenfalls innerhalb der äußeren Prozedur erklärt ist.

Da für jeden Term ein 16-bit-Wort verwendet wird und Bit 0 zur Unterscheidung der beiden Terme dient, kann der Wert der Distanzadresse und damit die Länge einer lokalen Schachtel in Worten nicht größer als 32767 sein.

5.5 Benutzer-Schnittstellen

Bedienmöglichkeiten, Protokollaufbau und Umfang der Fehlerdiagnostik entsprechen etwa dem, was von Siemens in den Übersetzern für die größeren Anlagen des Systems 300 (330 und R-Maschinen) realisiert wurde.

Es gibt ca. 20 Bedienparameter, wie z.B.

DATEIEN / DATENTRÄGER

- Quelldatei
- Zieldatei
- Datenträger für %INCLUDE-Module
- " " Arbeitsdatei

ZAHLENWERTE FÜR

- Satzlänge der Quelle
- Zeilenlänge des Protokolls
- Anzahl Zeilen /Seite des Protokolls

BINÄRE WERTE (JA/NEIN)

- Protokoll des Quellprogramms
- Codegenerierung (Nein: Nur Syntaxanalyse)
- Feldgrenzenprüfung im Zielprogramm
- Crossreferenzliste
- IL1 - Protokoll
- Verschiedene Diagnostikinformationen für Wartungszwecke

Das Protokoll enthält auf dem Titelblatt die für den aktuellen Übersetzungsprozeß maßgeblichen Werte aller Bedienparameter. Jede Protokollseite erhält eine Kopfzeile mit Angabe der Übersetzerversion und Datum und Uhrzeit der Übersetzung. Das Protokoll des Quellprogramms wird mit Zeilennummer insgesamt und Zeilennummer innerhalb eines %INCLUDE-Moduls und mit der Blockschachtelungstiefe der jeweiligen Zeile versehen.

Fehlermeldungen erscheinen im Anschluß an das Quellsprache-protokoll im Klartext mit Verweis auf die Nummer der fehlerhaften Zeile. Etwa 500 mögliche unterschiedliche Fehlertexte erlauben eine recht genaue Lokalisierung aufgetretener Fehler.

5.6 Technische Daten

Im folgenden seien kurz die wichtigsten technischen Daten des Übersetzers angegeben.

Hauptspeicherbedarf	25 K Worte
Arbeitsform auf Peripherspeicher	400 K Worte
Arbeitsdatei " "	250 K Worte

Übersetzungsgeschwindigkeit
(ohne Drucken des Quellspracheprotokolls)

Nur Syntaxanalyse:	ca. 5 Zeilen /s
Zielprogramm in Grundsprache:	ca. 3 "
" " Assemblersprache:	ca. 2 "

Verlängerungsfaktoren gegenüber
"guten" Assemblerprogrammen

Programmcode:	ca. 2
Ausführungszeit:	ca. 2

6. Hilfsmittel zur Handhabung

Um die Installation und Benutzung des Übersetzers zu erleichtern, wurden einige Hilfsmittel geschaffen, die für 310-Anwender von allgemeinem Interesse sind.

6.1 Verbund zum Großrechner Siemens 7760

Editieren und Archivieren der Quellprogramme erfolgt im IITB vorwiegend auf einem Großrechner Siemens 7760, der unter BS2000 mit ca. 40 Terminals vorwiegend im Dialog betrieben wird. Mit einer modifizierten Blattschreiber-Ausschaltung der 310 an einer Terminal-Schnittstelle (V24) der 7760 wurde ein einfacher

Datenverbund realisiert, der folgende Funktionen ermöglicht:

- Kommando-Übergabe an Großrechner
- Editieren auf dem Großrechner
- Überspielen von Dateien in beiden Richtungen

6.2 Datenaustausch mit Siemens 330 und R30

Die auf einem Großrechner IBM 370 begonnene Entwicklung des Übersetzers wird vom Lieferer inzwischen überwiegend auf einer Siemens 330 betrieben. Zur Erleichterung des Austauschs von Plattenkassetten mit dieser und einer im Herbst 1979 im IITB installierten R30 wurde ein Programm geschrieben, welches das Umkopieren zwischen Listen von physikalisch adressierten Sektorbereichen und 310-Dateien ermöglicht.

Zur Datensicherung und Reorganisation stark zerstückelter Platten ermöglicht dasselbe Programm auch ein schnelles dateiweises Umkopieren, und zwar entweder

- alle Dateien einer eingegebenen Liste oder
- alle Dateien mit Ausnahme der in einer Liste genannten.

6.3 Monitor MP10

Der Monitor MP10 ermöglicht nicht nur die Übergabe eines in einer Datei abgelegten Kommandostroms an monitorabhängige Programme (MAP), sondern auch das Vor- und Rückwärtsverzweigen innerhalb des Kommandostroms als programmierte Reaktion auf Meldungen der MAP. Er erleichtert vor allem die Installation des Übersetzers: sie dauert mit MP1Ø und LB1Ø auf der 310 nur ca. 3 Minuten gegenüber 9 Minuten mit BD3Ø und ORG-Lader auf einer R3Ø.

7. Weiterentwicklung

In den vorangegangenen Kapiteln wurde ausschließlich über erprobte und seit längerem in Anwendung befindliche Eigenschaften des Übersetzungssystems berichtet. Abschließend sei ein Ausblick auf teils noch in Arbeit befindliche, teils fertiggestellte, aber noch nicht eingesetzte Weiterentwicklungen gegeben.

7.1 Optimierung des Übersetzers

Auf IL1-Ebene und damit unabhängig vom jeweiligen Zielrechner werden folgende Optimierungen vorgenommen:

- Elimination gemeinsamer Teilausdrücke in Basisblöcken, insbesondere
- nur einmalige Berechnung der Adresse von Feld- oder Strukturkomponenten, wenn diese im Basisblock mehrfach benutzt werden,
- Berechnung der Adresse von Feldkomponenten mit konstantem Index soweit möglich zur Übersetzungszeit,
- Auflösung logischer Ausdrücke in Folgen von Test- und Sprungbefehlen.

Im Codegenerator, d.h. maschinenspezifisch erfolgen z.B. folgende Verbesserungen:

- Halten der Basisadresse statischer Vorgänge in Registern, soweit möglich,
- dynamische Initialisierung von Variablen durch Direktkonstante, bei FIXED(4)-Konstanten durch RC-Befehle.

7.2 PEARL-Binder

Der in Arbeit befindliche PEARL-Binder PB10 wird folgende Funktionen realisieren:

- Ausgabe der gebundenen Programme in verschieblicher Grundsprache
- Prüfung der Verträglichkeit von Deklaration und Spezifikation Globaler Objekte. Dazu gibt der Übersetzer zusätzlich zur Grundsprache noch eine Beschreibung mit deren Typattributen aus.
- Ermittlung des Platzbedarfs im Laufzeitkeller (ähnlich BD30)
- Erstellen einer Tabelle von Inter-Stationsreferenzen des RDC-Systems (zur Auflösung durch den RDC-Lader).

Literatur

/1/ Heger, D.; Steusloff, H.; Syrbe, M.: Echtzeitrechnersystem mit verteilten Mikroprozessoren. Forschungsbericht DV 79-01, Bundesministerium für Forschung und Technologie, 1979.

/2/ Heine, P.: PEARL-Programmierung auf der SIEMENS 310, Betriebssystem. 11. Jahrestagung des SAK 1, KfA Jülich, 1980.

/3/ Kippe, J.: PEARL-Programmsysteme zur Lösung umfangreicher Automatisierungsaufgaben.
11. Jahrestagung des SAK 1, KfA Jülich, 1980.

/4/ Grimm, R.; Hellriegel, W.; Laubsch, H.; Rudolf, M.; Sassenhof, A.; Syrbe, M.: Bildprogrammierbares Ein-/Ausgabe-Farbbildschirmsystem (EAF) als Warte - Grundprinzipien, Realisierung, Erprobung. Kernforschungszentrum Karlsruhe GmbH, KfK-PDV 134, 1978.

/5/ Werum, W.; Windauer, H.: PEARL. Process and Experiment Automation Language. Verlag Vieweg, Braunschweig, 1978.

/6/ DIN 66253 Teil 1: Programmiersprache PEARL, Basic PEARL Normentwurf, Juni 1978.

/7/ DIN 66253 Teil 2: Programmiersprache PEARL, Full PEARL Entwurf zur Normvorlage, April 1980.

/8/ Steusloff, H.: Zur Programmierung von räumlich verteilten dezentralen Prozeßrechensystemen. Dissertation Universität Karlsruhe, 1977.

Fraunhofer-Institut
für Informations-und
Datenverarbeitung · IITB

75 Karlsruhe, den 08.05.1980

PEARL-Programmsystem zur Lösung umfangreicher Automatisierungsaufgaben

J. Kippe, Karlsruhe

1. Einleitung

Die in diesem Beitrag zu besprechenden Automatisierungsvorhaben nehmen im Rahmen der Tagung in gewisser Weise eine Sonderstellung ein, da sie Erfahrungen behandeln, die nur indirekt auf die Anwendung von SIEMENS-Prozeßrechnern übertragen werden können. Die Programmerstellung bedient sich des PEARL-Übersetzungssystems auf einer 310 K /1/, die erstellten PEARL-Programme laufen jedoch auf durch das IITB entwickelten Mikrocomputern ab, sogenannten RDC-Mikroprozessorstationen. Diese sind jedoch vom Befehlsumfang mit der 310 kompatibel. Darüberhinaus ist das Grundkonzept des RDC-Mikroprozessorsystems weitgehend unabhängig vom eingesetzten Rechnertyp. Die zu besprechenden Programmstrukturen können somit auch von allgemeinem Interesse sein. Im übrigen ist inzwischen ein PEARL-Betriebs- und Laufzeitsystem für die 310 einsatzbereit./2/ Es fehlte jedoch die Zeit, die im folgenden beschriebenen Einsatzfälle auch auf diesen Maschinen nachzuspielen.

Nach einigen einleitenden Bemerkungen soll ein kurzer Überblick über die eingesetzten Rechner und die Anlagenkonfiguration gegeben werden. Der Vorstellung der Hardware schließt sich im zweiten Teil die Beschreibung der Programmstruktur an. Den Abschluß bildet die Diskussion der Erfahrungen beim praktischen PEARL-Einsatz.

Die Programmiersprache PEARL (Process Experiment Automation Realtime Language) wurde in den Jahren 1970 bis 1976 gemeinsam von Industrie und Hochschulen mit erheblicher Förderung des BMFT entwickelt. Das Ziel dieser Bemühung war, durch die Verwendung einer Echtzeitsprache die Softwareproduktions- und -wartungs-

kosten erheblich zu senken, sowie ein Hilfsmittel zu entwickeln, das die Erstellung mehrfach verwendbarer (portabler) Software erlaubt. Im IITB wird seit Mitte 1978 das portable PEARL-Übersetzungssystem der Firma MBP/Werum praktisch eingesetzt, zunächst auf einer SIEMENS 310 K und seit Ende letzten Jahres auch auf einer SIEMENS R 30.

Über die Realisierung von zwei Automatisierungsprojekten im Bereich der Stahlerzeugung mit Hilfe der Programmiersprache PEARL und die dabei gewonnenen Erfahrungen soll hier berichtet werden.

Es handelt sich dabei einmal um die Automatisierung einer Tiefofenanlage, bei der 28 RDC-Mikroprozessorstationen zum Einsatz kommen sollen. Diese Anlage ist in einer ersten Ausbaustufe seit gut einem Jahr in Betrieb. Das zweite Projekt stellt die automatische Steuerung einer Legierungsanlage dar. Hier sollen 6 RDC-Mikroprozessorstationen zum Einsatz kommen. Das System läuft zur Zeit im Testbetrieb und wird in Kürze ausgeliefert.

2. Anlagenkonfiguration

Die RDC-Mikroprozessorstationen bilden ein fehlertolerantes, räumlich verteiltes Prozeßrechnersystem (RDC-System, RDC $\hat{=}$ Really Distributed Controlsystem), das durch einen Lichtleiterring gekoppelt ist (Bild 1). Dabei übernehmen die Mikroprozessorstationen die Regelung und Steuerung der Teilprozesse, ihre Programmierung erfolgt in PEARL.

Alle RDC-Stationen sind gleichartig strukturiert. Sie enthalten jeweils 2 Mikroprozessoren des Typs SAB 3000, die den Prozeßmikroprozessor (PμP) und den Leitungsmikroprozessor (LμP) bilden. Über einen Busschalter können die Prozessoren auch auf den Speicherbereich des jeweils anderen Prozessors sowie auf den Ein/Ausgabebus zugreifen.

Darüberhinaus sind im Fehlerfall durch geeignete elektronische Schalter, die über Zustandsüberwacher gesteuert werden, der Prozeßmikroprozessor, der Leitungsmikroprozessor mit Sender und Empfänger und die Prozeßsignalenein/ausgabe voneinander trennbar. Damit sind Fehler tolerierbar.

Insbesondere bleibt so die Ein/Ausgabe auch dann steuerbar, wenn der PµP ausfällt, nämlich über die Sammelleitung durch den LµP. Diese Betriebsart heißt Fremdsteuerung. Wählt man eine doppelte Prozeßein/ausgabe, so läßt sich auch der Totalausfall einer Station durch eine andere Station abfangen. Der gesamte Adreßbereich einer Station beträgt 64 K Worte à 16 bit. Davon entfallen auf den Arbeitsspeicher des Prozeßmikroprozessors 48 K, der Rest ist durch den Leitungsmikroprozessor, die Prozeßsignalein/ausgabe sowie durch Spezialregister belegt.

Dem Anwender zur Programmierung zugänglich ist nur der Prozeßmikroprozessor, er ist vom Befehlssatz her kompatibel mit den SIEMENS-Maschinen der Serie 3ØØ. Seine Programmierung erfolgt in PEARL.

Während die RDC-Stationen die Steuerung des Prozesses übernehmen, dienen zwei in das RDC-System integrierte 31Ø-Rechner als EAF-System (Ein/Ausgabe-Farbbildschirm-System) und als Programmier- bzw. Ladesystem.

Das EAF-System erlaubt die Prozeßbedienung mit virtuellen, an die jeweilige Situation anpaßbaren Tastaturen mit Lichtgriffelbedienung. Es stellt die zentrale Warte des Gesamtsystems dar. Hier erfolgt die Zusammenfassung aller Meßwerte und ihre Darstellung als Zeichen, Fließbilder, Balken oder Kurven in mehrfacher Bildschirmgröße, die durch Bildrollen intuitiv zugänglich ist.

Die an jedem Rechner vorhandenen Stationsbedienfelder bilden die zweite, dezentrale Anzeige- und Bedienebene. Sie erfüllen alle für die Prozeßführung notwendigen Funktionen wie Soll- und Grenzwertänderungen, Schalten, Stellen sowie die Anzeige aller

Meßwerte und Schaltzustände. Bei Ausfall der zentralen Warte läßt sich der Prozeß auf diese Weise weiterfahren durch Bedienung vor Ort.

3. Programmstruktur

Das gesamte Programmsystem läßt sich in Funktionsgruppen gliedern, denen die einzelnen PEARL-Moduln entsprechen (Bild 2). Um eine konsequente Modularisierung und damit eine optimale Entkopplung der Teilfunktionen zu erreichen, wurde ein Listen- und Prozedurmodul eingeführt, der als zentrale Schnittstelle fungiert. Die Listen bilden auch die Schnittstelle zu anderen RDC-Stationen und dem EAF-System.

Die Erfassung der Meßwerte erfolgt sowohl zyklisch als auch ereignisgesteuert. Sie werden durch den Ein/Ausgabemodul aufbereitet und in die Meßwertlisten des Listen- und Prozedurmoduls eingetragen. Die Ausgabe von Schaltbefehlen an den Prozeß kann in gleicher Weise zyklisch bzw. ereignisgesteuert erfolgen. Der Datenfluß verläuft hier von den Listen über den Ein/Ausgabe-Modul an die Ausgabebaugruppen.

Über die Meßwertlisten werden die Meßwerte und Prozeßzustände allen anderen Moduln zur Verfügung gestellt. Die Struktur dieser Listen wird im vorliegenden Fall durch das EAF-System bestimmt, das auf diese Listen direkt zugreift und daraus die Bildschirmdarstellung ableitet.

Dieser direkte Zugriff kann als Quasi-DMA-Verkehr verstanden werden und wird durch das Betriebssystem und den Leitungsmikroprozessor abgewickelt (Globalkommunikation). Die zweite Kommunikationsform innerhalb des Systems ist die sogenannte Intertaskkommunikation. Sie erfordert eine sendende und eine empfangsbereite Task. Das wird durch die Kommunikationsmoduln realisiert. Durch Intertaskkommunikation erfolgt der Austausch von Softwarestatusmeldungen, Meldungs- und Alarmtelegramme, Schaltbefehlen und Sollwertvorgaben.

Der Stationsbedienfeldmodul erlaubt die Anzeige einzelner Listenelemente und die Veränderung durch Schaltbefehle oder Sollwertänderungen. Durch Auswertung der Bedieneingaben wird ein bestimmtes Listenelement adressiert. Die zyklische Ausgabe bringt die angewählten Objekte auf einer Siebensegmentanzeige zur Darstellung.

Die zentralen Prozeduren dienen insbesondere dem schreibenden Zugriff auf die Meßwertlisten. Diese Art der Listeneintragung wurde aus zwei Gründen gewählt.

1. Da die Meßwertlisten als ausgelagerter Teil des EAF-Systems verstanden werden können, sind beim Zugriff die Konventionen des EAF-Systems zu beachten. Hierzu gehören insbesondere die Grenzwertüberwachung und die Erzeugung von Alarmmeldungen.

2. Für die noch zu besprechende Rekonfiguration ist das Führen einer Kopie der Meßwertliste in einer Ausweich- oder Ersatzstation notwendig. Die Aktualisierung dieser Liste wird durch die Prozeduren automatisch vorgenommen.

Die als Anlauf- und Rekonfigurationssteuerung bezeichnete Funktionsgruppe wickelt den Start eines Programmsystems ab. Dazu gehört insbesondere die Initialisierung und Verzeigerung von Listen sowie der Anstoß von Statusmeldungen an die anderen Stationen.

Speziell die Rekonfigurationssteuerung bearbeitet Ausfälle und Wiederinbetriebnahmen anderer Stationen. Sie übernimmt den geordneten Start und die geordnete Abschaltung von Moduln, die für die Übernahme der ausgefallenen Funktionen anderer Stationen vorgesehen sind. Zur geordneten Abschaltung gehört insbesondere die Freigabe belegter Betriebsmittel vor der Terminierung des Moduls.

Den reinen Anwenderprogrammen obliegt die Steuerung und Regelung der technischen Prozesse. Sie laufen in einer durch die oben beschriebenen Moduln realisierten Umgebung ab. Die Informationen über den Prozeßzustand entnehmen sie den Meßwertlisten, ermittelte Reaktionen werden mittels der schon angesprochenen Prozeduren wiederum in die Listen eingetragen.

Es ist wesentlich, daß bei der Erstellung der Steuerungs- und Regelprogramme durch den Anwender den Besonderheiten des verteilten Systems kaum Beachtung zu schenken ist. Die Konvention des Listenzugriffs paßt das Anwendungsprogramm automatisch an den aktuellen Systemzustand und an die eventuellen Ausfälle einzelner Stationen an.

3.1 Rekonfigurationskonzept

Im Normalbetrieb steuert jede Mikroprozessorstation ihren eigenen Prozeß. Zur Einführung fehlertoleranter Eigenschaften wird nun bei Ausfall einer Station bzw. bei Ausfall eines Teilbereichs einer Station eine Rekonfiguration des Systems vorgenommen. Dazu werden die Stationen paarweise zusammengefaßt. Bei Ausfall einer Station übernimmt die jeweilige Nachbarstation als Ausweichstation die ausgefallenen Funktionen mit.Je nach Ausführung der Anlage kann entweder über eine zweite Prozeß-ein/ausgabe oder über den Leitungsmikroprozessor auf den Nachbarprozess zugegriffen werden.

Als Reaktion auf den Ausfall einer Hardwarekomponente gibt der Leitungsmikroprozessor jeder Station des Systems eine Meldung über den Hardwarezustand. Diese Hardwarestatusmeldungen verursachen einen Interrupt in jedem Prozeßmikroprozessor und den Start eines als lokaler Beobachter bezeichneten Systemprogramms. Der lokale Beobachter startet oder beendet PEARL-Moduln in Abhängigkeit vom Hardwarezustand des Gesamtsystems. Um bei der Übernahme der Funktion einer ausgefallen Nachbarstation einen möglichst reibungslosen Übergang zu gewährleisten, ist den Anwenderprogrammen, also den Regelungs- und Steuerungsprogrammen bei Rekonfigurationsanlauf der aktuelle Zustand des Prozesses zur Verfügung zu stellen. Das bedeutet, daß beim Anlauf auf dem Ausweichrechner bei Rekonfiguration, bzw. bei Anlauf auf dem regulären Rechner nach Rückkonfiguration die aktuellen Meßwertlisten vorhanden sein müssen (Wiederaufsetzen).

Aus diesem Grunde verfügt jede beteiligte Station über zwei Meßwertlisten,die reguläre Liste (für den eigenen Prozeß) und die Ausweichliste (für den Nachbarprozeß). Die Verwaltung dieser Listen erfolgt nach dem Benutzerprinzip, d.h. jeder

Prozeß ist für seine eigenen Listen verantwortlich.

Bei Rekonfiguration, d.h. bei Ausfall der regulären Station, soll der Ausweichprozeß in der Nachbarstation gestartet werden. Da nun aber die aktuellen Listen in der regulären Station nicht mehr erreichbar sind, wird bei EAF-Anlauf die gesamte Liste in die Ausweichstation kopiert und dann im Normalbetrieb ständig auf dem aktuellen Stand gehalten. Bei der Rückkonfiguration liegen die Verhältnisse einfacher, da die Listen in der Ausweichstation zur Verfügung stehen. Der anlaufende reguläre Prozeß braucht die Ausweichliste lediglich in die regulären Listen zu kopieren.

Bei diesem Verfahren der Ausfallsicherung findet also keine Übernahme durch ein back-up System statt, die Ausweichstation übernimmt neben ihrem eigenen Prozeß zusätzlich den Prozeß der Nachbarstation.

Wenn die zu bedienenden Prozesse gleichartig sind, läßt sich die funktionsbeteiligte Redundanz durch mehrfache Benutzung des Programmcodes in Form von globalen, reentranten Prozeduren ausnutzen. Der zusätzliche Programmaufwand für die Rekonfiguration beschränkt sich somit auf die Formulierung von Tasks, die den Aufruf und die Versorgung dieser Prozeduren beinhalten.

4. Erfahrungen

Bei der Codierung des vorgestellten Programmsystems zeigten sich die wesentlichen Vorteile von PEARL. Neben den von anderen höheren Programmiersprachen bekannten Vorteilen (wie geringer Schreibaufwand, leichte Lesbarkeit, Selbstdokumentation) zeigte sich vor allem der Nutzen der in PEARL erstmalig verfügbaren Sprachmittel zur Einplanung und Koordinierung von parallel ablaufenden Programmen. Ferner ist die durch eine blockstrukturierte Sprache wie PEARL gegebene hohe Programmiersicherheit hervorzuheben.

Der vorhandene Sprachumfang läßt kaum Wünsche offen.
Als Mangel ist vielleicht das Fehlen einer der EQUIVALENZ-Anweisung in FORTRAN vergleichbaren Anweisung anzusehen.
So ist die Überlagerung von Datenstrukturen unterschiedlichen Typs nur durch die vom Compiler tolerierte Typeninkosistenz bei der Einstellung von Referenzvariablen zu erreichen.Sehr praktisch und effizient erwies sich die Manipulation komplexer Datenstrukturen, insbesondere die Zuweisung ganzer Strukturen, die sehr speicherplatz- und laufzeitgünstig ist.

In einer für die Prozeßdatenverarbeitung konzipierten Echtzeitsprache sollen Standard-Operatoren für die Übergabe der aktuellen Uhrzeit und des Datums an das Programm zur Verfügung stehen. Ihr Fehlen kann man wohl als Kuriosität betrachten.

Die bereits angesprochenen Sonderfunktionen der Mehrrechnerprogrammierung (Globalkommunikation, Intertaskkommunikation Fremdsteuerung) sind erst teilweise auf PEARL-Ebene realisiert. Die Bedienung der Prozeßein/ausgabe anderer Stationen hauptsächlich für die Betriebsart Fremdsteuerung steht durch die Einführung globaler DATIONs voll zur Verfügung. Das gleiche gilt für globale Interrupts. Durch Angabe einer Stationsnummer im Systemteil unterscheidet sich die Behandlung der Fremd-E/A für den PEARL-Programmierer nicht von der der eigenen Ein/Ausgabe.

Die Kommunikationsfunktionen Globalkommunikation und Intertaskkommunikation sind zur Zeit durch den Aufruf von Betriebssystemprozeduren realisiert.

Bei der Intertaskkommunikation ist als Zieladresse die Angabe einer Stationsnummer und einer Kennziffer (Port-Nummer) für die Empfangstask erforderlich.

Die Globalkommunikation, d.h. der Zugriff zu gemeinsamen Objekten in anderen Stationen, erfordert die Auflösung der Adreßbezüge. Diese erfolgt gegenwärtig durch Intertaskkommunikation. Um diese Handhabung für den Anwender zu erleichtern, können die Global-

objekte über ein Identifikationssystem angesprochen werden. Prozeduren des globalen Prozedurmoduls übernehmen die dynamische Verwaltung in Abhängigkeit vom Rekonfigurationszustand des Gesamtsystems.

Die Implementation eines Mehrrechner-PEARL-Systems ist vorgesehen, in dem neben der globalen Prozeßein/ausgabe und der impliziten Globalkommunikation durch Benutzung gemeinsamer Datenobjekte auch explizite Intertaskkommunikation (Botschaftensystem) möglich ist. Hier ist die Verwendung globaler Kommunikations-DATIONs als Nachrichtenübergabestellen und READ/WRITE für das Empfangen und Senden von Nachrichten geplant.

Ein Programmsystem der vorgestellten Art umfaßt etwa 10 000 Programmzeilen Quelltext. Das sind etwa 35 Tasks, 25 globale Prozeduren sowie Meßwertlisten, sie belegen zusammen etwa 40 K Speicherplatz. Hinzu kommen dann nochmals ca. 7 K für das PEARL-Betriebssystem und -Laufzeitsystem. Die Erstellungszeit betrug ungefähr 2 Mannjahre.

Vergleiche zwischen PEARL-codierten und assembler-codierten Programmen hinsichtlich Speicherplatz- und Laufzeitbedarfs sind bisher nur punktuell durchgeführt worden. Der Speicher-Mehrbedarf für compilierte Programme kann danach mit etwa 100% angegeben werden.

Allerdings ist auch die unkritische Nutzung komplexer Sprachmittel, abhängig von der Abbildbarkeit auf den Maschinenbefehlsvorrat, in hohem Maße für die schlechte Speicherplatz- und Laufzeiteffizienz verantwortlich. Ein typisches Beispiel ist die Einzelbitadressierung in komplexen Datenstrukturen, der sogenannte Bitselektor. Dieser wird von Laufzeitsystem-Routinen durchgeführt, die den allgemeinsten Fall dieser sehr mächtigen Sprachfunktion berücksichtigen müssen.

Eine Verbesserung der Laufzeit- und Speicherplatzsituation durch Compileroptimierung sowohl auf Zwischensprach-Ebene als auch im Codegenerator ist hier jedoch zu erwarten.

Die modulare Struktur des vorgestellten Programmsystems stellte sich als außerordentlich testfreundlich heraus. Da die Listen und Prozeduren die zentrale Schnittstelle bilden, lassen sich die Moduln nahezu unabhängig voneinander behandeln.

Die Test- und Inbetriebsetzungsphase zeigt, das Programmierer mit Assemblererfahrungen sehr bald versuchen, Korrekturen in ihrem übersetzten und gebundenen Programm durchzuführen. Das führt nicht nur zu Änderungen an den betrachteten Statements, sondern eventuell auch zu Eingriffen in die vom Compiler oder Betriebssystem angelegten und verwalteten Listen. Das steht natürlich in krassem Widerspruch zum Bestreben, ein Programm sicher und zuverlässig zu machen. Es darf jedoch nicht außer acht gelassen werden, daß mindestens während der Inbetriebnahme gewisse Konzessionen gemacht werden müssen.

Unter diesem Gesichtspunkt wäre die Ergänzung des PEARL-Programmiersystems durch ein komfortables Testsystem von großem Nutzen. Die bisher erst zur Hälfte nutzbare Sprache PEARL würde dadurch handlicher und dem Programmierer würde das Leben erleichtert.

5. Literatur

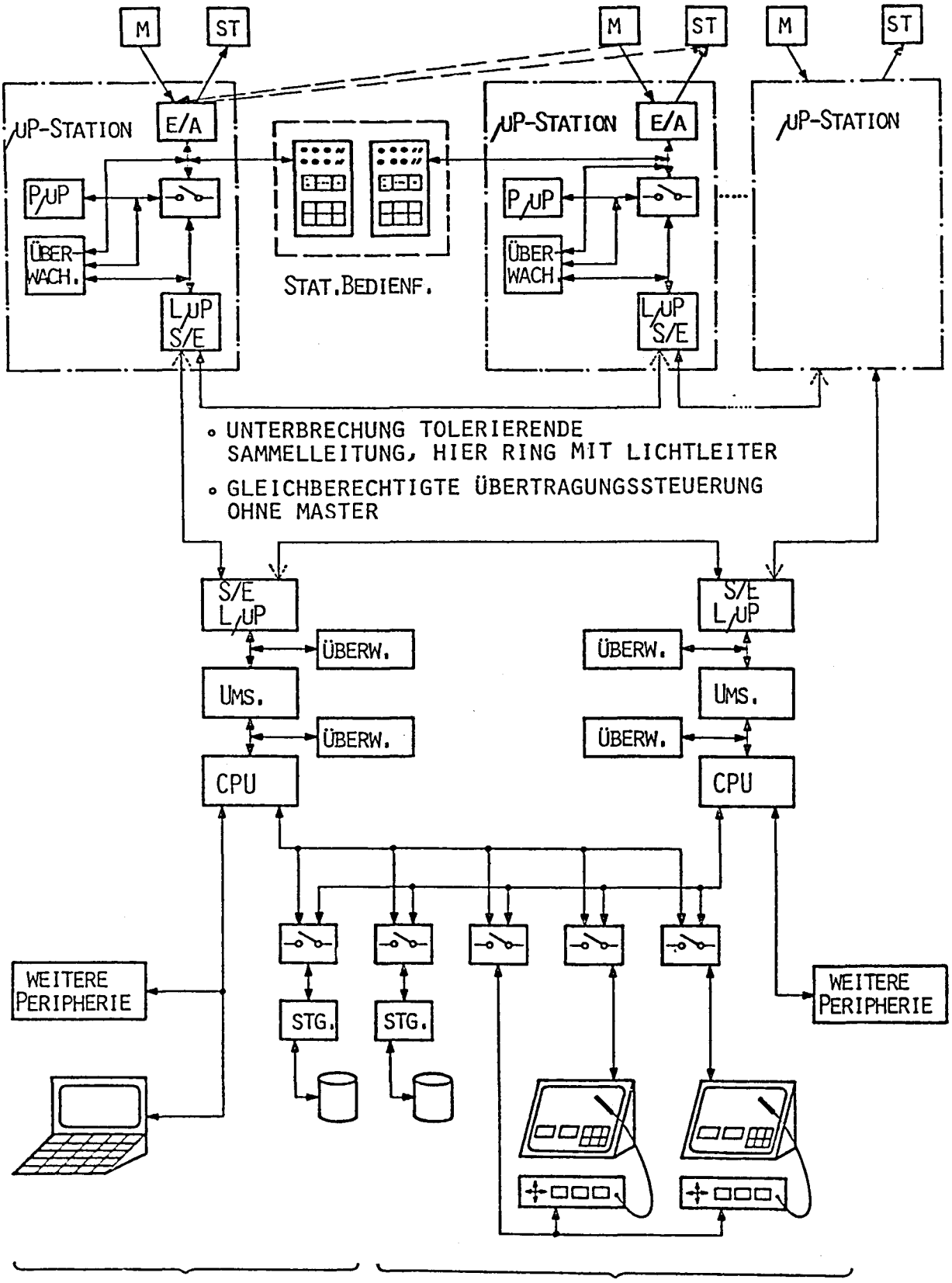
/1/ L. Lorenz: PEARL-Programmierung auf der SIEMENS 31Ø: Übersetzungssystem. Siehe vorliegender Tagungsband.

/2/ P. Heine: PEARL-Programmierung auf der SIEMENS 31Ø: Betriebssystem. Siehe vorliegender Tagungsband.

/3/ D. Heger, H. Steusloff, M. Syrbe: Echtzeitrechnersystem mit verteilten Mikroprozessoren. Forschungsbericht BMFT-FB DV 79-Ø1.

/4/ H. Steusloff: Zur Programmierung von räumlich verteilten, dezentralen Prozeßrechnersystemen. Dissertation, Universität Karlsruhe.

/5/ G. Bonn, W. Heil, J. Kippe, F. Sänger: Selbsttest und Selbstkonfiguration von Prozeßrechnersystemen am Beispiel des RDC-Systems. IITB-Mitteilungen 1979.


M
ST
µP-STATION
E/A
P/µP
ÜBER-WACH.
L/µP S/E
STAT.BEDIENF.
ÜBER WACH
S/E L/µP
UNTERBRECHUNG TOLERIERENDE SAMMELLEITUNG, HIER RING MIT LICHTLEITER
GLEICHBERECHTIGTE ÜBERTRAGUNGSSTEUERUNG OHNE MASTER
ÜBERW.
UMS.
CPU
WEITERE PERIPHERIE
STG.
DOKUMENTATIONS- UND PROGRAMMIERBEDIENPLATZ
PROZESS-BEDIENPLÄTZE

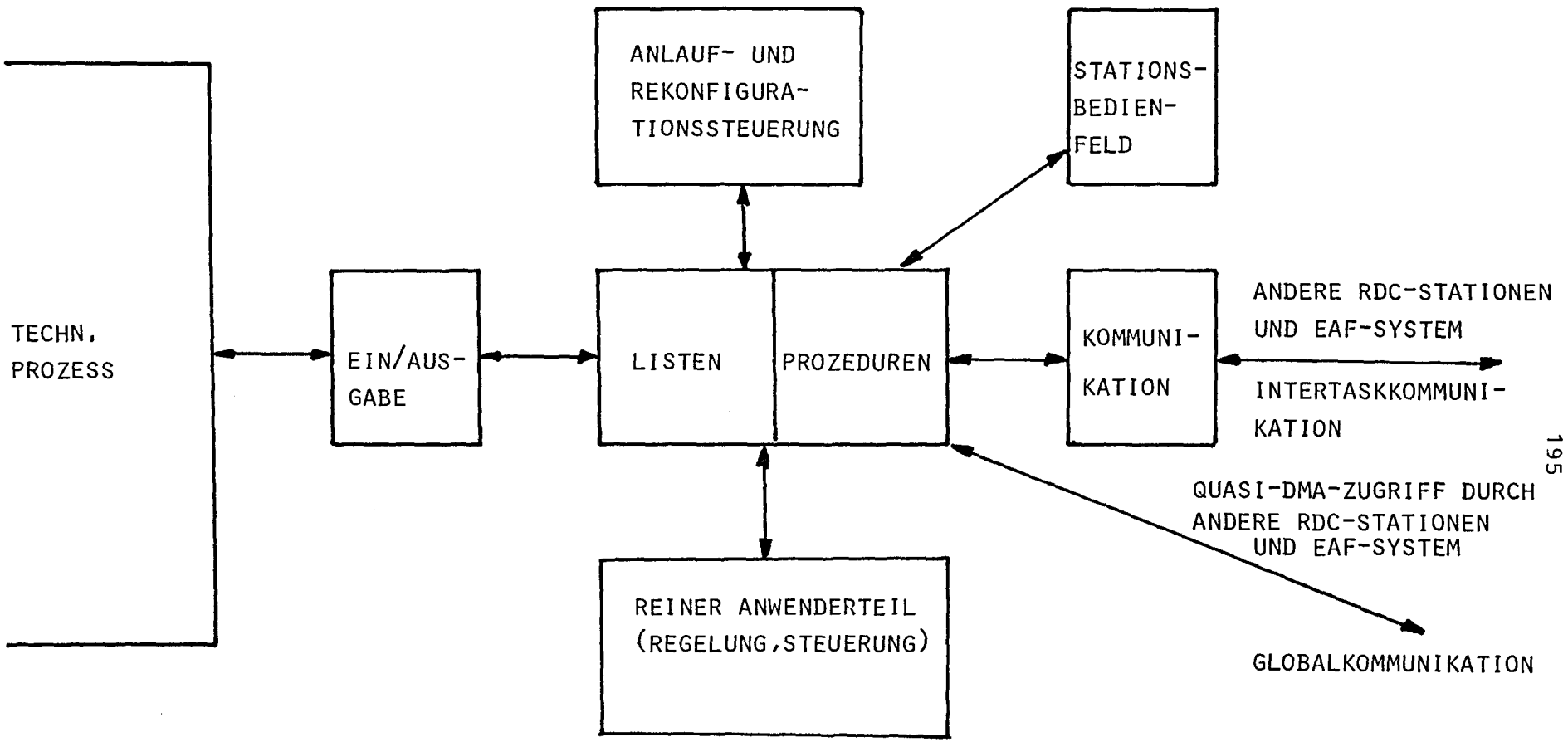

ANLAUF- UND
REKONFIGURA-
TIONSSTEUERUNG
STATIONS-
BEDIEN-
FELD
TECHN.
PROZESS
EIN/AUS-
GABE
LISTEN
PROZEDUREN
KOMMUNI-
KATION
ANDERE RDC-STATIONEN
UND EAF-SYSTEM
INTERTASKKOMMUNI-
KATION
QUASI-DMA-ZUGRIFF DURCH
ANDERE RDC-STATIONEN
UND EAF-SYSTEM
REINER ANWENDERTEIL
(REGELUNG,STEUERUNG)
GLOBALKOMMUNIKATION

Bild 2:

Ein PEARL-Cross-Compiliersystem für eine 310

und Anwendungen in der Labormeßtechnik

M. Trautner, R. Besold

Physikalisches Institut der Universität Erlangen

Ein PEARL-Cross-Compiliersystem für eine 31Ø und Anwendungen in der Labormeßtechnik

Das Cross-Compiliersystem wurde von der Gruppe Datenverarbeitung des Physikalischen Instituts der Universität Erlangen, gefördert durch BMFT/PDV, entwickelt [1]. Ein wesentlicher Aspekt dabei war die Erprobung portabler PEARL-System-Software [2]. Der Aufwand für die Erstellung des gesamten Systems konnte hierdurch gegenüber den vergleichbaren Komponenten einer früheren Implementation für die 3Ø6 [3] von 6 auf 2,5 Mannjahre gesenkt werden.

1. Das Übersetzungssystem

PEARL-Quellmodule werden auf der 3Ø6 vom sog. Oberen Compilerteil [4] in die Zwischensprache CIMIC/1 [5], diese von einem Codegenerator in 3ØØ-Assembler und dann vom Cross-Assembler ASSB in Grundsprache übersetzt. Über Teletype-Kopplung werden die Grundsprachdateien von der 3Ø6 zur 31Ø auf Floppy-Disk überspielt. Ein Hauptmodul und eventuell Prozedurmoduln werden dort durch den Ladebinder LB1Ø mit den benötigten Elementen einer PEARL-Laufzeitbibliothek zu einem Programm gebunden, dessen Start und Ablauf durch ein PEARL-Betriebssystem gesteuert wird.

2. Kurzbeschreibung der neu erstellten Systemkomponenten

2.1 Codegenerator

Der Codegenerator [6] besteht aus vier Läufen, die zusammen 25% der Übersetzungszeit benötigen (Oberer Compilerteil: 50%, Assembler: 25%). Aufgabe der einzelnen Läufe:

a.) Umsetzung von CIMIC/1 in einen Zifferncode
b.) Eigentliche Code-Erzeugung (Mehrere Dateien)
c.) Binden von Dateien mit Zeichen-Codeumsetzung (von Siemens-Intercode nach ASCII)
d.) Erzeugung eines Stackmoduls; dabei werden für jede Task der Gesamtbedarf an lokalen Daten errechnet, der bestimmt wird durch den Eigen-

bedarf und den Bedarf der von ihr aufgerufenen Prozeduren.

2.2. Dateiübertragung 3Ø6-31Ø

Die Grundsprachedateien werden über eine Teletypeleitung mit 4800 baud übertragen; hardware-Schnittstelle bei der 3Ø6 ist ein CAMAC-TTY-Einschub, bei der 31Ø eine Sichtgeräteanschaltung im TTY-Betrieb. Die Kopplungsprogramme sind jeweils in PEARL geschrieben und führen die Übertragung in Blöcken nach einem Protokoll gemäß DIN 66Ø19 durch, wobei fehlerhafte Übertragungen erkannt und nach negativer Quittung wiederholt werden können.

2.3. Das PEARL-Betriebssystem (PBS)

Das PBS [7][8] verwaltet die Ein/Ausgabe bei den "langsamen" E/A-Geräten, bearbeitet die Tasking- und Semaphoroperationen (durch Ein- und Ausketten von Tasks in Warteschlangen und Retten der Registertafel) und übernimmt die Erkennung und Bearbeitung von 16 Interrupts die über eine alarmbildende Digitaleingabe erzeugt werden. Es erlaubt beliebig viele Tasks, die beim Aktivierungsaufruf mit einer aktuellen Priorität versehen werden können.

Das PBS ist ein Anwendeprogramm (unter ORG-Verwaltung) und belegt die Programmebenen 13 und 14. Aufträge von PEARL-Tasks (alle Ebene 15) und Rückmeldungen nach einem ORG-Aufruf laufen über eine einzige Schnittstelle (Koordinierungszählererhöhung). Das PBS besteht jeweils zur Hälfte aus einem rechnerunabhängigen Verwaltungsteil (in einem virtuellen höheren Assembler geschrieben) und aus einem anlagenabhängigen Teil (in AS3ØØ formuliert), Gesamtlänge: 18ØØ Speicherworte.

2.4 Laufzeitsystem

Die Prozeduren des LZ-Systems sind reentrant formuliert, sodaß jede Prozedur - wenn überhaupt - nur einmal im Hauptspeicher stehen muß, auch wenn sie von mehreren Tasks gleichzeitig aufgerufen wird. Die so gewonnene Reduzierung des Speicherplatzbedarfs überwiegt bei weitem die

gegenüber nichtreentrant formulierten Prozeduren gering erhöhte Laufzeit.
Denplatzmäßig größten Teil des LZ-Pakets stellen die Formatierungs-Routinen für die Standard-E/A [9] dar (~ 7K Worte), die in PEARL formuliert und somit anlagenunabhängig sind. Alle anderen LZ-Prozeduren (Prozess-E/A, Zeichenkettenverarbeitung, Schnittstellenanpassroutinen, Arithmetische Routinen u.ä.) sind in Assembler geschrieben und belegen zusammen nur etwa 1,4K Worte.

3. Sprachumfang

Der für die 31Ø verwirklichte Sprachumfang ist, bis auf das Fehlen der graphischen EA vergleichbar mit dem früher für die 3Ø6 verwirklichten PEARL-Subset [10] (gleicher Compileroberteil). Letzterer wiederum entspricht funktionell (also von syntaktischen Unterschieden abgesehen) dem DIN-Normvorschlag für BASIC-PEARL [11], lediglich Strukturen sind noch nicht eingeführt.

Besonderheiten der Implementation:

Anzahl der Tasks beliebig
grundsätzlich reentrantfähige Anwenderprozeduren
getrennte Übersetzbarkeit von Hauptmodul (mit Tasks und Prozeduren)
und Prozedurmoduln (nur Prozeduren)

4. Der Einsatz des PEARL-Systems in der Labormeßtechnik

Abgesehen von der Verwendung im System selbst (in PEARL-formulierte Standard-E/A, Kopplungsprogramm, Programm zur Erstellung von Programmbibliotheken, Testprogramme) wird das PEARL-System für die 31Ø erfolgreich in der Labormeßtechnik eingesetzt.

4.1. Meßwerterfassung und Auswertung bei Zugversuch

Zur Erfassung und Reihenuntersuchung der Elastizitätskennlinien verschiedener Werkstoffe im Siemens-Forschungszentrum Erlangen diente (bis März 1980) eine 31Ø, die von einer Zugprüfmaschine über Analog- und Digitaleingabe Meßwerte übernahm. Einem hierfür bereits vorhandenen BASIC-Programm, das mit 0,6 sec pro Meßpunkt nicht die gewünschte Aufnahmerate (Folge: mangelnde Genauigkeit) erreichte, wurde ein PEARL-Programm gegenübergestellt. Dieses war nicht nur in der Datenerfassung wesentlich schneller (75 msec/Meßpunkt), sondern konnte außerdem parallel dazu - in einer anderen PEARL-Task - die gemessene Kurve auswerten, analysieren und bei ungewöhnlichem Verlauf die Messung abbrechen.

4.2. Aufnahme eines Molekülspektrums [12]

Das Experiment:

Ein Molekülstrahl wird von einem Elektronenstrahl mit einstellbarer Energie (von 5eV bis 5keV) senkrecht durchsetzt. Dabei werden die Moleküle in verschiedenster Weise angeregt (Vibration, Rotation, Dissoziation) und können hierauf Licht aussenden. Aus dem Spektrum, d.h. der Intensitätsverteilung des Lichts in Abhängigkeit von der Wellenlänge, lassen sich Schlüsse über den Aufbau des Moleküls ziehen.

Aufnahmeelektronik:

Die Erfassung des Spektrums erfolgt durch Messen der Intensität für je einen kleinen Wellenlängenbereich. Dazu dient ein Monochromator, der über ein drehbares Reflexionsgitter einen bestimmten Wellenlängenbereich des einfallenden Lichts auf seinen Ausgang abbildet. Die Intensität des Lichts - genauer: die Anzahl der einzelnen γ-Quanten- wird über einen Photomultiplier von einem Impulszähler gemessen. Die Winkelstellung des Gitters und damit die Lage des Wellenlängenbereichs wird über einen Schrittmotor gesteuert.

Anschlußstellen an die 31Ø:

Der Schrittmotor wird über eine Digitalausgabe gesteuert. Der Zähler wird über eine kombinierte Digitalein/ausgabe bedient, wobei entweder eine bestimmte Meßzeit oder eine Impulszahl vorgegeben und die gemessene Impulszahl (bzw. benötigte Zeit) gelesen wird. Schrittmotor und Zähler liefern Signale (insbes. Fertigmeldungen), die als Interrupts über eine weitere (alarmbildende) Digitaleingabe von Programm verarbeitet werden.

Das Programm:
Das PEARL-Programm mißt nach entsprechenden Bedienungsangaben durch schrittweise Steuerung des Schrittmotors und Lesen des Zählers das Spektrum, gibt die Zählraten auf Schnelldrucker aus und speichert sie außerdem (zusammen mit Kontrolldaten) für weitere Auswertungen auf Floppy-Disk. Programmlänge: 55Ø Quellzeilen, bzw. 15K Speicherworte (davon 8K Laufzeitprozeduren).

Literaturverzeichnis

(1) P. Holleczek et al.: PEARL Cross-Compiliersystem für den Prozeßrechner Siemens 31Ø; Regelungstechnische Praxis 1979, Heft 12.

(2) P. Holleczek: Erfahrungen mit portabler Systemsoftware bei PEARL-Implementationen für die Prozeßrechner Siemens 3Ø6 und 31Ø; Portable Software, Tagung I/1980 des German Chapter of ACM, H.J. Schneider (Hrsg.) B.G. Teubner, Stuttgart 1980.

(3) P. Elzer, P. Holleczek: ASME-Compiler wird der Öffentlichkeit vorgestellt. Regelungstechnik 12 (1975).

(4) B.Eichenauer: Entwicklungskonzept für ein Programmsystem zur Erstellung und Erzeugung von Systemsoftware für die Prozeßautomation; PDV-Mitteilungen 1/1971; Ges. f. Kernforschung mbH, Karlsruhe.

(5) Mühlhahn: Spezifikation CIMIC/1, Bericht KFK-PDV 75, Ges. f. Kernforschung mbH, Karlsruhe 1976.

(6) M. Trautner; P. Holleczek: Spezifikation von Codegenerator und Laufzeitfunktionen der 31Ø-PEARL-Implementation; PDV-Interner Bericht, 1979.

(7) R. Rössler: Betriebssystemstrategien zur Bewältigung von Zeitproblemen in der Prozeßautomatisierung; Disseration, Universität Stuttgart 1979.

(8) A. Fleischmann: Anpassung eines rechnerunabhängigen PEARL-Betriebssystems für die Siemens 31Ø, Studienarbeit am IMMD, Universität Erlangen 1979.

(9) K. Lampe: Erstellung der Standard-EA-Laufzeitroutinen des PEARL-ASME-Stufe-1-Compiliersystems in PEARL; Diplomarbeit, IMMD Universität Erlangen.

(10) Gruber, Inderst, Piche: Programmieranleitung für das ASME-1-PEARL-SUBSET; Bericht KFK-PDV 100, Ges. f. Kernforschung mbH, 1976.

(11) Programmiersprache PEARL, Basic PEARL, Entwurf DIN 66253, 1978.

(12) R. Besold: Diplomarbeit am Physikalischen Institut der Universität Erlangen, 1980.

D A I M L E R - B E N Z - A G Abt: E7BA : BIB: DDP,MOBEVT

Dokumentation des Vortrages "MOBEST" am 12.5.80
auf der Tagung des Siemens-Anwender-Kreises SAK1
Ort: Kernforschungs-Zentrum Juelich

Zusammenfassung:

MOBEST ist ein fernbedienbarer Monitor (REMOTE JOB MONITOR), fuer die Prozessrechner-Serie Siemens-Systeme 300/16 Bit, mit dem es moeglich ist, JOB-Laeufe von dezentral aufgestellten Bedien-Geraeten aus zu aktivieren.

Bei der Aktivierung mehrerer JOB's werden die JOB's in einer monitor-internen JOB-Warteschlange gepuffert.

Ueber die Angabe einer JOB-Klasse wird bei der Aktivierung eines JOB's indirekt eine maximale JOB-Laufzeit, d.h. die Start-Stop-Zeit des JOB's vorgegeben. Ist der JOB nach Ablauf dieser Zeit noch nicht beendet, so wird der JOB durch den Monitor abgebrochen und damit zwangsbeendet.

Bei Eintrag mehrerer JOB-Auftraege in der JOB-Warteschlange werden JOB's aus niederigeren Klassen vor JOB's aus hoeheren Klassen unabhaengig von der zeitlichen Aktivierungsreihenfolge bearbeitet.

Technische Daten des Programms:

Hardware	: ZE 330 / R10 R20 / R30 R40
Software	: ORG 330 K / ORG 300 P / ORG 300 PV
Benutzte Software	: BIBEAS, KOMI und SPOOL 300
Simulationsstufe	: SIMKON, SIM30R bei ORG 330 K
Programmart	: Hauptprogramm (linear)
Sprache	: ASS300 und MAS300
Speicherbedarf	: ca 23 k-Worte

Problemstellung

Die Standard-Monitore MONITOR330 und MONITOR300V sind bedienbare Anwenderprogramme, die mit Unterstuetzung der Betriebssysteme, den ORG's den automatischen Ablauf von Einzel-Programmen, den sogenannten MAP's. d.h. monitorabhaengigen Programmen, steuern und ueberwachen.

Die notwendigen Steuerinformationen fuer diese Aufgaben lesen die Standard-Monitore vorzugsweise ueber den Lochkarten-Leser, aber auch aus Quellsprach-Bibliotheks-Elementen von Externspeichern, bei letzteren erfolgt jedoch bei Element-Ende eine automatische Umschaltung zum Lochkarten-Leser.

Bei logischen Ablauffehlern innerhalb eines JOB's schalten die Monitore bei Eingabe der Steuerinformationen aus Bibliotheks-Elementen ebenso auf Lochkarten-Leser um. In bestimmten Fehlerfaellen arbeiten die Monitore gar nicht weiter, sondern brechen die JOB-Bearbeitung ab und warten auf Operator-Bedienung.
Ausserdem werden die JOB's streng sequentiell abgearbeitet, d.h. eine Prioritierung von JOB's ist nicht vorgesehen, ferner ist eine Pufferung von JOB-Auftraegen nicht moeglich.

Die Standard-Monitore haben keinerlei Kontrolle ueber die Ausgabe-Aktivitaeten der MAP's, die Monitore koennen somit nicht gewaehrleisten, dass alle zu einem JOB gehoerenden Protokoll-Ausgaben "zusammenhaengend" auf einem Ausgabegeraet ausgegeben werden.

Dieses Betriebsverhalten der Standard-Monitore fuehrt bei dezentralisierter Programm-Erstellung mittels Bildschirm-Terminals, z.B. Datensichtstation 3974 und Einsatz von Editoren, z.B. EDISP oder MEDIS zu folgenden Auswirkungen:

- Es kann bei der dezentralen JOB-Auftragserteilung von einem Terminal aus, immer nur ein JOB aktiviert werden, simultane weitere JOB-Aktivierungen weisen die Monitore ab.

- Bei der zentralen Aktivierung von JOB's im Rechenzentrum mittels Lochkarten-Leser ist jedoch zwingend ein Operator erforderlich.
  Zusaetzlich entsteht in diesem Fall das Problem der Auftragsuebermittlung vom Benutzer an den Operator.

- Der Benutzer hat vom dezentralen Terminal aus keinerlei Informationsmoeglichkeit ueber den Zustand seines JOB's.

- Die Protokoll-Ausgaben der MAP'S koennen auf mehrere Protokoll-Geraete verteilt sein, bzw. sind evtl. in verschiedenen Text-Bibliotheken und/oder Bibliotheks-Elementen hinterlegt.

Es bedarf dann einer weiteren Organisation, die einzelnen Text-Ausgaben wieder zusammenzufuehren und dem Benutzer geschlossen zur Verfuegung zu stellen.

Problem-Loesung

Es muss ein Monitor-Programm erstellt werden, welches folgenden Bedingungen gerecht wird:

- Aktivierung von JOB's ausschliesslich durch Bedienung von beliebigen Bediengeraeten zu beliebigen Zeitpunkten.
- Automatische Pufferung von JOB-Auftraegen bei schon taetigem Monitor, auch ueber Neustart und Wiederanlauf des Systems.
- Ausnahmslos Verwendung von Bibliotheks-Elementen fuer die JOB-Steuerinformationen, dies sind Lochkarten-Aequivalente, deshalb im folgenden auch einfach Steuerkarten genannt.
- Weitestgehende Kompatibilitaet mit den JOB-Steuerkarten der Standard-Monitore, um bereits bestehende JOB-Steuerkarten mit den geringstmoeglichen Aenderungen uebernehmen zu koennen.
- Zwangweises Umlenken aller Text- und Protokoll-Ausgaben aus einem JOB auf ein einziges Geraet.

Loesungs-Konzept

Die oben aufgestellten Forderungen werden von einem zweigeteiltem Programm erfuellt.

Ein Bedienteil, der die staendige Bedienbarkeit sichert und damit die Schnittstelle zun Benutzer darstellt, im folgenden "MONBED" genannt und
einem Steuerteil, der die angebotenen JOB-Steuerkarten abarbeitet, im folgenden "MONSTP" genannt.

Diese beiden Programmteile verkehren ueber eine Datenschnitt-stelle, die JOB-Warteschlange, miteinander.
In diese JOB-Warteschlange werden bei der JOB-Auftrags-Bedienung von einem Terminal aus, durch MONBED Eintraege gemacht und damit an MONSTP zur Bearbeitung uebergeben.

Treffen waehrend der Bearbeitung eines JOB-Auftrages durch MONSTP weitere JOB-Auftrags-Bedienungen ein, so werden sie von MONBED ebenfalls in die JOB-Warteschlange eingetragen und danach von MONSTP sequentiell abgearbeitet.

Abgearbeitete JOB's werden von MONSTP in die JOB-Ergebnisliste eingetragen. In dieser Liste werden sowohl die JOB-Auftragsparameter als auch die Start- und Ende-Zeit des JOB's, sowie ein evtl. vorhandener Fehlercode im Falle eines JOB-Abbruches. Diese Daten stehen damit fuer spaetere JOB-Auskuenfte noch zur Verfuegung.

Beide Listen sind im Programm selbst realisiert und haben jeweils Platz fuer 100 Eintraege, damit ist einerseits eine genuegend grosse Puffermoeglichkeit vorhanden, andererseits ist eine schnelle Verarbeitung der Daten gewaehrleistet, da keinerlei Plattenzugriffe notwendig sind.

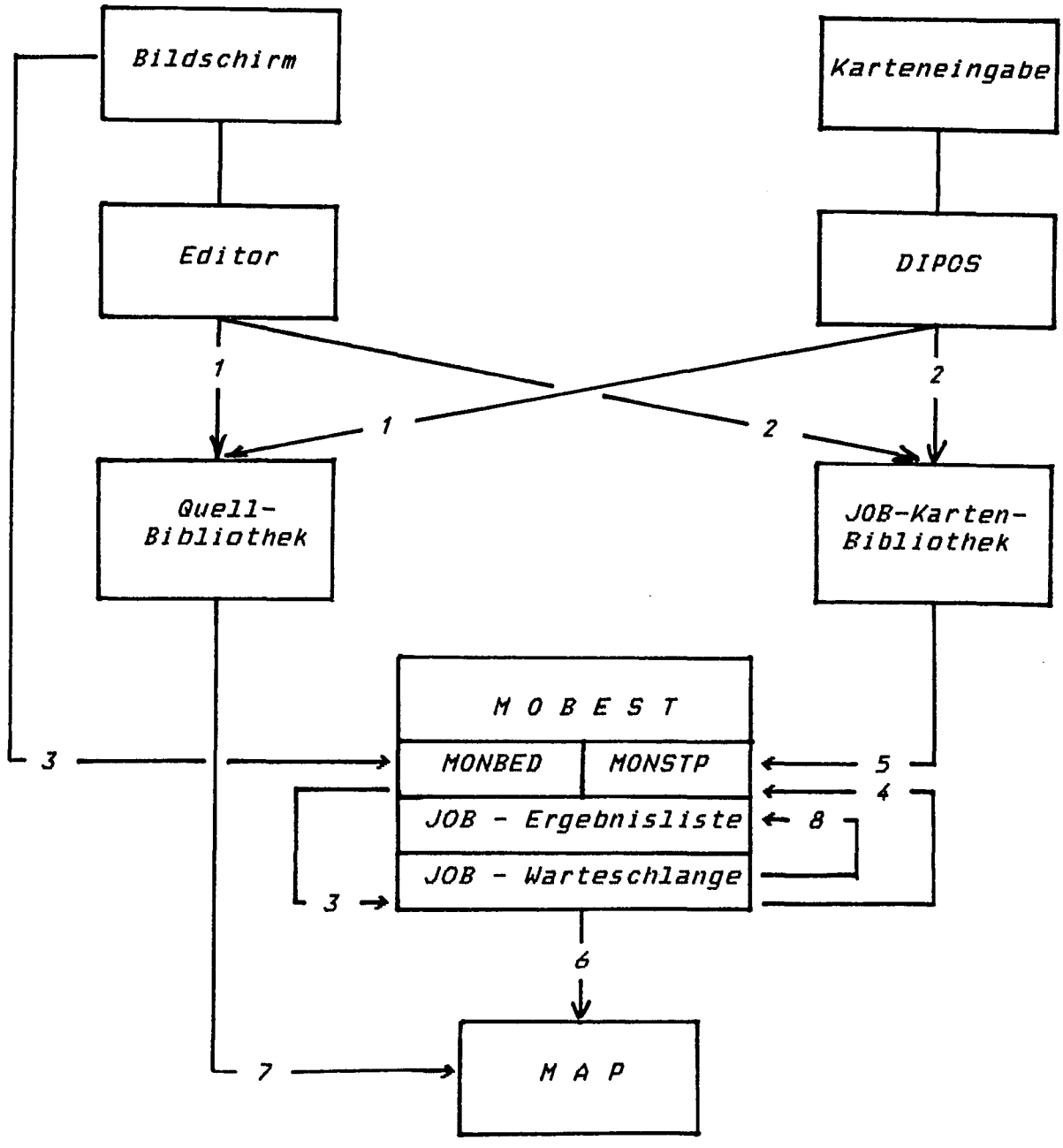
Die JOB-Warteschlange kann durch eine LV-Angabe beim Laden des Monitors vergroessert werden, sodass je nach der Groesse dieser Angabe wesentlich mehr JOB-Auftraege gepuffert werden koennen.
Hinweis: Beim Einsatz des Monitors unter ORG 300 PV wird der V-Teil Verschnitt automatisch fuer die JOB-Warteschlangen-Verlaengerung ausgenutzt.
Die Job-Ergebnisliste hat eine statische Laenge von 100 Plaetzen und ist als Umlaufpuffer organisiert, sodass jeweils zeitlich die letzten 100 bearbeiteten JOB's vermerkt werden koennen.

Fuer den Fall dass die Laenge der JOB-Ergebnisliste fuer die gewuenschten Auskuenfte ueber schon bearbeitete JOB's nicht ausreichend ist oder die JOB-Bearbeitungs-Daten laengerfristig fuer statistische Zwecke, z.B. Rechenzentrums-Kosten-Abrechnung, gespeichert werden muessen, ist der Anschluss fuer ein weiterverarbeitendes Programm vorgesehen.

Zur Uebersicht ueber alle Komponenten und der Arbeitsweise siehe Bild 1 (naechste Seite).

D A I M L E R - B E N Z - A G Abt: E7BA : BIB: DDP,MOBEVT

Bild 1



Erlaeuterungen zu Bild 1 :

1. Der Benutzer erstellt bzw. korrigiert seine Quellsprache bzw. seine Daten mittels Editor oder DIPOS .

2. Der Benutzer erstellt bzw. korrigiert seine JOB-Steuerkarten mittels Editor oder DIPOS.

3. Der Benutzer gibt seinen JOB-Auftrag an den Bedienteil MONBED, der Auftrag wird von MONBED in die JOB-Warteschlange eingetragen und damit an den Steuerteil MONSTP uebergeben.

4. Der Steuerteil MONSTP liest den JOB-Auftrag aus der JOB-Warteschlange.

5. Der Steuerteil MONSTP bearbeitet die angebotenen JOB-Steuerkarten.

6. Der Steuerteil startet und/oder steuert und bedient das bzw. sequentiell die MAP's.

7. Das MAP bearbeitet gemaess der vom Steuerteil uebergebenen Bedienkarten die Quellsprache bzw. die Daten.

8. Nach JOB-Ende uebertraegt MONSTP den JOB aus der JOB-Warteschlange in die JOB-Ergebnisliste.
Ist der Anschluss fuer ein die JOB-Daten weiterverarbeitendes Programm aktiviert, so uebergibt der Steuerteil MONSTP diese Daten per Koordinierungszaehler-Start an dieses Programm.

Funktionelle Aenderungen gegenueber den Standard-Monitoren

Im Gegensatz zu den Standard-Monitoren sind allgemeine Funktionen, wie Laden, Starten und Loeschen von Programmen im Monitor MOBEST nicht realisiert.
Es sind nur die zur Steuerung von MAP's, d.h. von im System vorhandenen bzw. geladenen Programmen, notwendigen Funktionen realisiert, diese sind jedoch teilweise fuer den Rechenzentrums-Betrieb sinnvoll erweitert worden.
Mit Hilfe der neu eingefuehrten MAP-Steuerkarten //STBE, //BED und //ENBE, sowie der auch in neueren Versionen der Standard-Monitore vorhandenen Steuerkarten //MULTIMAP und //SINGLMAP ist es moeglich, das Standard-Bedien-Programm SBP als MAP zu betreiben, damit ist es auch mit dem Monitor MOBEST moeglich andere Programme im Sinne der allgemeinen Funktionen zu hantieren. Darueber hinaus sind natuerlich auch alle anderen Funktionen des SBP's mit Hilfe des Monitors an-

sprechbar.
Hinweis: Waehrend das SBP dem Monitor als MAP unterstellt ist, kann es nicht mehr von "Hand" bedient werden.

Die JOB's werden in sieben bzw. in acht Zeitklassen eingeteilt, der Benutzer muss bei der JOB-Auftragserteilung an MONBED eine Zeit-klasse angeben, in welcher der JOB laufen soll.
Der Steuerteil MONSTP bearbeitet nun bei gleichzeitigem Eintrag mehrerer JOB'S in verschiedenen Zeit-Klassen in der JOB-Warte-schlange grundsaetzlich die Auftraege in niederigeren Zeit-Klassen vor Auftraegen in hoeheren Zeit-Klassen, unabhaengig vom Zeitpunkt der Auftragserteilung. D.h. es werden JOB'S mit kuerzeren Laufzeiten vor den JOB's mit laengeren Laufzeiten ausgefuehrt.
Innerhalb einer Zeit-Klasse werden die JOB's nach zeitlicher Auf-tragsreihenfolge bearbeitet.

Die jeweilige Laufzeit der JOB's wird gemaess der vorgegebenen Zeit-Klasse ueberwacht, bei einer Laufzeit-Ueberschreitung wird der JOB durch den Monitor abgebrochen und der naechste JOB aus der niederigsten Zeit-Klasse, sofern vorhanden, bearbeitet.

Durch diesen Mechanismus ist es moeglich bei hoher Monitor- bzw. Rechner-Belastung kurze bzw. schnell laufende JOB's vorzuziehen und JOB's mit langen Laufzeiten in Zeiten niederigerer Rechner-Belastung (vorzugsweise Nachts) zu verschieben.

Um JOB's erst zu bestimmten Zeiten ausfuehren zu koennen, wurden eine Grenz-Zeit-Klasse und Grenz-Zeiten eingefuehrt.
Hiermit ist es moeglich, JOB's, deren zugeordnete Zeit-Klasse groesser oder gleich der Grenz-Zeit-Klasse ist, nur in dem durch die Grenz-Zeiten definierten Zeitraum auszufuehren, hierbei ist insbesondere daran gedacht, JOB's mit sehr langen Laufzeiten bzw. JOB's deren Ergebnisse nicht unmittelbar benoetigt werden, in die Nacht zu verschieben.

Bei der JOB-Auftragserteilung erhaelt jeder Auftrag vom Bedienteil MONBED eine JOB-Nummer zugeteilt, welche am bedienten Geraet an den Benutzer ausgegeben wird.
Mittels dieser Nummer oder auch des JOB-Namens (Elementname der JOB-Steuerkarten) kann der Benutzer zu einem spaeteren Zeitpunkt den Zustand seines JOB's abfragen, oder sofern der JOB noch nicht ausgefuehrt wurde, wieder stornieren.

Wahlweise ist ferner der Anschluss eines Programms moeglich, welchem nach der JOB-Ausfuehrung die JOB-Daten uebergeben werden koennen. Dieses Programm kann dann z.B. diese Daten mit Mitteln der Daten-Verwaltungs-Systeme DATORG oder DVS 300 abspei-chern, um sie fuer statistische Zwecke verfuegbar zu machen.

Ferner ist im Monitor ein Anschluss an das SPOOL-System SPOOL 300

/12/ eingebaut. Damit ist es moeglich die gesamten Text-Ausgaben eines JOB's zunaechst auf einer Platte zwischenzuspeichern und die eigentliche Text-Ausgabe auf einem druckendem Geraet von der JOB-Ausfuehrung zu entkoppeln.
Hinweis:Dieser Anschluss kann nur dann benutzt werden, wenn der Monitor unter dem ORG 300 P oder ORG 300 PV laeuft, da nur in diesen ORG'S die Funktionen des SPOOL-Systems realisiert sind. Unter ORG 330 K wird der Versuch des Anschlusses an das SPOOL-System mit Fehler-Meldungen abgewiesen.

Teilnehmerliste

11. Jahrestagung des Siemens-
Prozeßrechner-Anwenderkreises 1
12. - 14. Mai 1980 KFA Jülich

Peter Abend	Hahn-Meitner-Institut für Kernforschung GmbH Glienicker Str. 100 1000 Berlin 39
Klaus Amft	Hoesch Hüttenwerke AG Postfach 902 4600 Dortmund 1
Heinz Ankermann	Siemens AG, ZN Dortmund Märkische Str. 14 4600 Dortmund
Prof.Dr. W. Aßmus	Fachhochschule Dortmund Sonnenstr. 100 4600 Dortmund
E. Bachner	Zentralabteilung Allgemeine Technologie ZAT Kernforschungsanlage Jülich Postfach 1913 517 Jülich
Ralph Barthelmeus	Siemens AG, Dynamowerk Nonnendammallee 1000 Berlin 13
Dr. Manfred Baum	RTWH Aachen Templergraben 66 5100 Aachen
Albrecht Benseler	Siemens AG Postfach 120 7000 Stuttgart 1
Franz Benzinger	Siemens AG ZN München VE6 Richard-Strauß-Str. 76 8000 München 80

Helmut Bergmann	Siemens AG ZN Essen Kruppstr. 16 4300 Essen
Peter Bernards	Siemens AG Rheinbrückenstr. 50 7500 Karlsruhe
Gottfried Bonn	Fraunhofer Gesellschaft 75 Karlsruhe
Eduard Breimann	Bayer AG IN PLT PST Geb.: G8 4047 Dormagen
Dr.-Ing. Hermann Buseck	Bundesanstalt für Straßenwesen Brühlerstr. 1 5000 Köln 51
Wilhelm Corpus	Siemens AG E65E3 Frauenauracher Str. 8520 Frankfurt
Jochen Diemer	Siemens AG ZN Stuttgart VE6 Postfach 120 7000 Stuttgart 1
H. Ent	Daimler-Benz-AG Stuttgart-Ut Postfach 7000 Stuttgart-Ut.
Dipl.-Ing. G. Fischer	Fachhochschule Dortmund Sonnenstr. 100 4600 Dortmund
Peter Fischer	Siemens, Abt. E 364 Rheinbrückenstr. 50 7500 Karlsruhe
Walter Fischer	Universität Erlangen Techn. Chemie I Egerlandstr. 3 8520 Erlangen
Dieter Flemming	Messerschmitt Bölkow Blohm (MBB) 8012 Ottobrunn

Willi Förtig	Siemens AG E364 Rheinbrückenstr. 51 7500 Karlsruhe 21
Norbert Gerber	Amt für Nachrichtenwesen der Bundeswehr Wilhelmstr. 87 5483 Bad Neuahrweil
B. Glismann	Siemens AG Lindenplatz 1 2000 Hamburg 1
Jürgen Gropp	Bayer AG Abt. IN PLT PST 3 415 Krefeld 11
Dr. Hans Grundner	Messerschmitt Bölkow Blohm (MBB) 8012 Ottobrunn
Wolfram Guhr	Universität Dortmund Abteilung Chemietechnik Postfach 500500 4600 Dortmund 50
Wilfried Haaf	Berufsförderungswerk Heidelberg Bonhöfferst. 6900 Heidelberg-Wieblingen
Gerhard Hauptmann	Berufsförderungswerk Heidelberg Banhöfferstr. 6900 Heidelberg-Wieblingen
Klaus Heim	Universität Karlsruhe Rechenzentrum Postfach 6380 75 Karlsruhe
Dr. Peter Heine	Fraunhofer-Institut für Informations- und Datenverarbeitung Sebastian-Kneipp-Str. 12-14 7500 Karlsruhe 1
Margit Heinze	Inst. f. Steuerungstechnik Seidenstr. 36 7000 Stuttgart 1
Hermann Josef Hellmich	Rheinisch-Westfälisches Elektrizitätswerk AG Hauptschaltleitung Brauweiler 5024 Pulheim

Heinz Hochmuth	Dornier System GmbH Postfach 1360 7990 Friedrichshafen
Edgar Hochrein	Fichtel & Sachs AG Abt. Techn. Rechenzentrum Postfach 1140 8720 Schweinfurt
Rainer Höhmann	Jenaer Glaswerke Schott u. Gen. Abtlg. TDE-3 Postfach 2480 6500 Mainz
H. Huppertz	Zentralabteilung für Elektronik Kernforschungsanlage Jülich Postfach 1913 517 Jülich
Heinz Inhoven	Zentralabteilung Forschungsreaktoren Kernforschungsanlage Jülich Postfach 1913 517 Jülich
W. Janssen	Zentralabteilung für Elektronik Kernforschungsanlage Jülich Postfach 1913 5170 Jülich
Hans-Peter Jünemann	Siemens AG ZN Hamburg, VE-Aut Lindenpl. 2 2000 Hamburg 1
Eginhard Jungmann	Siemens AG, ZFA PTE 3 Aut 11 Geisenhausenerstr. 18 8000 München 70
Gerhard Jüst	Siemens AG Rheinbrückenstr. 50 7500 Karlsruhe
Erwin Kammler	Rheinisch-Westfälisches Elektrizitätswerk AG Hauptschaltleitung Brauweiler 5024 Pulheim
Aloys Kampmann	Kraftwerk-Union AG Postfach 1142 4330 Mülheim a.d. Ruhr

Wilh. Kämpkes	Lehrstuhl für Anlagentechnik Postfach 500500 4600 Dortmund 50
Rolf Kerpe	Kernforschungszentrum Karlsruhe Institut für Datenverarbeitung in der Technik Postfach 3640 7500 Karlsruhe
Herrn Kevekordes	Univ. GH Paderborn FB 17 Warburger Str. 100 4790 Paderborn
Dr. Eberhard Kienzle	Fachhochschule f. Tecnik Kanalstr. 33 7300 Esslingen
B. Kinnen	Siemens AG, E 362 8520 Erlangen
Jörg Kippe	Fraunhofer-Institut f. Informations- u. Datenverarbeitung Sebastian-Kneipp-Str. 12-14 7500 Karlsruhe
Robert Klein	Forschungsinstitut für Anthropotechnik FAT Königstr. 2 5307 Wachtberg-Werthoven
Hans Klingler	Siemens AG, E 61 Rheinbrückenstr. 50 75 Karlsruhe 21
A. Klöcker	Zentralabteilung Allgemeine Technologie ZAT Kernforschfungsanlage Jülich Postfach 1913 517 Jülich
H.-J. Koch	Bayerische Motoren Werke AG Petuelring/Abt. AS-25 8000 München 40
Norbert Köhler	Hoesche Hüttenwerke AG Postfach 902 4600 Dortmund 1

Hellmut Kopec
Bayer AG
Abt. IN PLT PST 3

415 Krefeld 11

Herrn Krautner
Universität Erlangen

8520 Erlangen

Herrn Lauque
ESOC Europ. Space
Operations Centre
Robert-Bosch-Str. 5

6100 Darmstadt

Peter Leck
Erprobungsstelle 71
d. Bundeswehr
Berlinerstr. 115

2330 Eckernförde

J. Lerch
Zentralabteilung für Elektronik
Kernforschungsanlage Jülich
Postfach 1913

517 Jülich

Hartwig Lindemann
Physikalisch-Technische Bundesanstalt
Bundesallee 100

3300 Braunschweig

Prof. Friedrich Karl Lingemann
Universität Gesamthochschule
Paderborn Fachbereich Elektr.
Energie- und Maschinentechnik
Grüne Hecke 29

4770 Joest

Peter Lippold
DFVLR- Inst. chem. Antriebe
und Verfahrenstechnik
Hardthäuser Wald

7108 Hardthausen

Herr Loben
Siemens AG VE/FG 5
Franz-Geuer-Str. 10

5000 Köln 30

Lothar Lorenz
Fraunhofer-Institut für
Informations- und Datenverarbeitung
Sebastian-Kneipp-Str. 12-14

7500 Karlsruhe 1

Lothar Loup
Zentralinstitut für angewandte
Mathematik
Postfach 1913

5170 Jülich

Siegfried Lutz	ZN-München Richard-Strauß-Str. 76 8 MÜnchen 80
Eberhard Mangold	Siemens AG, ZN Stuttgart VE6 Postfach 120 7000 Stuttgart 1
Herr Meibrink	Siemens AG, VE/FG 5 Franz-Geuer-Str. 10 5000 Köln 30
Franz-Josef Molitor	Forschungsinstitut für Anthropotechnik FAT Königstr. 2 5307 Wachtberg-Werthoven
Christophorus Mosko	Fa. INFODAS Trompeterstr. 18 4018 Langenfeld
H. Neidmann	Daimler-Benz-AG Postfach 7000 Stuttgart-Ut.
Günther Noack	DFVLR-Inst. chem. Antriebe und Verfahrenstechnik Hardthäuser Wald 7108 Hardthausen
Sönke Paulsen	Erprobungsstelle 71 der Bundeswehr Berliner Str. 115 233 Eckernförde
Hartmut Peters	Zentralinstitut für angewandte Mathematik Kernforschungsanlage Jülich Postfach 1913 5170 Jülich
Rolf Pfeiffer	Siemens AG Postfach 130 7000 Stuttgart 1
Wilhelm Pfeiffer	Universität Gesamthochschule Paderborn Fachbereich Elektr. Energie- u. Maschinentechnik Grüne Hecke 29 4770 Joest

Albert Plewina	Physikalisch-Technische Bundesanstalt Bundesallee 100 3300 Braunschweig
Dr. Ernst-Eberhard Pohl	Ackerstr. 22 3300 Braunschweig
Franz-J. Polster	Kernforschungszentrum Karlsruhe Institut f. Datenverarbietung in der Technik Postfach 3640 7500 Karlsruhe
N. Portner	Siemens AG, E 362 8520 Erlangen
Prof.Dr.-Ing. Hanfried Prehn	Universität Gesamthochschule Paderborn, Fachbereich Elektr. Energie- u. Maschinentechnik Grüne Hecke 29 4770 Joest
Georg Preiss	Fichtel & Sachs AG Abt. Techn. Rechenzentrum Postfach 1140 8720 Schweinfurt
Karlheinz Preißler	Siemens AG Rheinbrückenstr. 50-52 7500 Karlsruhe
Peter Prinz	EDS-Systemtechnik An der Schurzelter Brücke 1 5100 Aachen
Rainer Punkte	Fachhochschule Münster Abt. Steinfurt Stegerwaldstr. 39 4430 Steinfurt
Siegfried Rackwitz	Amt für Nachrichtenwesen der Bundeswehr Wilhelmstr. 87 5483 Bad Neuahrweil
Dr. Ramolla	Hoechst AG Werk Bobingen Postfach 101627 89 Augsburg 1

Norbert Rasche	KraftwerkUnion AG Wiesenstr. 35 4330 Mülheim Ruhr
P. Reinhart	Zentralabteilung für Elektronik Kernforschungsanlage Jülich Postfach 1913 517 Jülich
Klaus Renner	Siemens AG Karlsruhe Abteilung E STE 34 Postfach 211262 7500 Karlsruhe 21
Georg Wolfgang Rexroth	Siemens AG ZN München VE6 Richard-Strauß-Str. 76 8000 München 80
Bernd Rivelsky	Jenaer Glaswerke Schott u. Gen. Abtlg. TDE-3 Postfach 2480 6500 Mainz
Wolfgang Rössler	Siemens AG Abt. FTE 3 AQT 223 Schertlinstr. 8 8000 München 70
F. Rongen	Zentralabteilung für Elektronik Kernforschungsanlage Jülich Postfach 1913 517 Jülich
Klaus-Jürgen Rost	Hoesche Hüttenwerke AG Postfach 902 46 Dortmund 1
D- Runggaldier	Forschungslabor FKS3 G.-Scharowsky-Str. 8520 Erlangen
Rainer Rybarsch	Physikalisch-Technische Bundesanstalt Bundesallee 100 3300 Braunschweig
Herr Sahle	Siemens AG, VE/FG5 Franz-Geuer-Str. 10 5000 Köln 30

Helga Sassen	Zentralabteilung Forschungsreaktoren Kernforschungsanlage Jülich Postfach 1913 5170 Jülich
Erhard Seemann	Siemens AG Ackerstr. 22 3300 Braunschweig
Lothar Sommer	Siemens AG Am Maschpark 1 3000 Hannover 1
Klaus Specker	EDS-Systemtechnik An der Schurzelter Brücke 1 5100 Aachen
Hans Schlosser	Lehrstuhl für Anlagentechnik Postfach 500500 4600 Dortmund 50
Willibald Schmaderer	Institut für Klinische Chemie Klinikum Großhadern der Universität München Bereich EDV Marchionistr. 15 8000 München 70
Rudolf Schmid	Siemens AG ZFA-WQA-MQ1 Schertlinstr. 8 8 München 70
Wolfgang Schmid	Inst. f.elektr. Maschinen Antriebe und Bahnen Hans Sommer Str. 65/66 3300 Braunschweig
Peter Schmidt	Berufsförderungswerk Heidelberg Banhöfferstr. 6900 Heidelberg-Wieblingen
Peter Schmidt	Fachhochschule Münster Stegerwaldstr. 39 443 Steinfurt
Klaus-Dieter Schmidtke	Fachhochschule Münster Stegerwaldstr. 39 4330 Steinfurt

Josef Schmitz	Kraftwerk Union AG TM-TRP Postfach 011420 4330 Mülheim/Ruhr
Volker Schreiber	Siemens Aktiengesellschaft Gutleutstr. 31 6000 Frankfurt/M. 1
Dr. Harald Schumny	Physikalisch-Technische Bundesanstalt Bundesallee 100 3300 Braunschweig
Dr. Hans-Joachim Schuster	Physikalisch-Technische Bundesanstalt Bundesallee 100 3300 Braunschweig
Ulrich Schwan	RWTH Aachen Templergraben 66 5100 Aachen
Manfred Stiller	IBAT 5000 Köln
A. Tack	Hahn-Meitner-Institut Glienicker Str. 1000 Berlin
Dr. W. Tenten	Zentralabteilung für Elektronik Kernforschungsanlage Jülich Postfach 1913 5170 Jülich
John Thorton	Inst. f. elektr. Maschinen Antriebe und Bahnen Hans Sommer Str. 65/66 3300 Braunschweig
Jürgen Vetter	Siemens AG, E 361 A- Ruppstr. 2 8520 Erlangen
Michel Veysseyre	Dornier System GmbH Postfach 1360 7990 Friedrichshafen
Frau Wagner	Siemens AG VE/FG5 Franz-Geuer-Str. 10 5000 Köln 30

Dr. Horst Weber	Battelle-Institut e.V. Am Römerhof 35 6 Frankfurt/Main 90
Prof.Dr. Klaus Weise	Physikalisch-Technische Bundesanstalt Bundesallee 100 3300 Braunschweig
Johann Weiss	TU Wien Inst. f. Datenverarbeitung Gußhausstr. 27 A-1040 Wien/Österreich
Herbert Widner	TU Wien Inst. f. Datenverarbeitung Gußhausstr. 27 A-1040 Wien/Österreich
Dr. Hans Windauer	Entwicklungsbüro Wulf Werum Glogauer Str. 2a 2120 Lüneburg
Erich Zartner	Rheinisch-Westfälisches Elektrizitätswerk AG Hauptschaltleitung Brauweiler 5024 Pulheim
Gernot Zöller	Rheinbrückenstr. 50 7500 Karlsruhe 21

Adressen des SAK - Vorstandes

Dipl.Ing. Peter Abend
Hahn-Meitner-Institut für Kernforschung Berlin GmbH Bereich D/M
Glienicker Straße 100

1000 Berlin 39

Tel.: 030/8009-2519

Dipl.Phys. Erhard Bachner
Kernforschungsanlage Jülich GmbH/ZAT
Postfach 19 13

5170 Jülich

Tel.: 02461/61-3284

Dipl.Math. Klaus Heim
Rechenzentrum der Universität Karlsruhe
Zirkel 2
Postfach 63 80

7500 Karlsruhe 1

Tel.: 0721/6082067

Dipl.Ing. Lothar Lorenz
Fraunhofer-Institut für Informations- und Datenverarbeitung
Sebastian-Kneipp-Str. 12-14

7500 Karlsruhe 1

Tel.: 0721/6091329

Dipl.Ing. Johann Weiss
Technische Universität Wien
Institut für Datenverarbeitung
Gußhausstr.27-29

A-1040 Wien

Tel.: 0043/222/657641-787

SAK-Sekretariat
Dipl.Ing. Peter Bernhards
Siemens AG -E612
Rheinbrückenstr. 50

7500 Karlsruhe 21

Tel.: 0721/595-2351