KfK-PDV 171
Juni 1979

PDV-Berichte

Industrielle Erfahrungen mit der Programmiersprache PEARL

T. Martin (Hrsg.)

Kernforschungszentrum Karlsruhe

PDV-Berichte

Die Kernforschungszentrum Karlsruhe GmbH koordiniert und betreut im Auftrag des Bundesministers für Forschung und Technologie das im Rahmen der Datenverarbeitungsprogramme der Bundesregierung geförderte Projekt Prozeßlenkung mit Datenverarbeitungsanlagen (PDV). Hierbei arbeitet sie eng mit Unternehmen der gewerblichen Wirtschaft und Einrichtungen der öffentlichen Hand zusammen. Als Projektträger gibt sie die Schriftenreihe PDV-Berichte heraus. Darin werden Entwicklungsunterlagen zur Verfügung gestellt, die einer raschen und breiteren Anwendung der Datenverarbeitung in der Prozeßlenkung dienen sollen.

Der vorliegende Bericht dokumentiert Kenntnisse und Ergebnisse, die im Projekt PDV gewonnen wurden.

Verantwortlich für den Inhalt sind die Autoren. Die Kernforschungszentrum Karlsruhe GmbH übernimmt keine Gewähr insbesondere für die Richtigkeit, Genauigkeit und Vollständigkeit der Angaben, sowie die Beachtung privater Rechte Dritter.

Druck und Verbreitung:

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640 7500 Karlsruhe 1

Bundesrepublik Deutschland

PDV

KfK-PDV 171

PROJEKT PROZESSLENKUNG MIT DV-ANLAGEN

FORSCHUNGSBERICHT KfK-PDV 171

INDUSTRIELLE ERFAHRUNGEN MIT DER PROGRAMMIERSPRACHE PEARL

VON

T. MARTIN (HRSG.)

KERNFORSCHUNGSZENTRUM KARLSRUHE GMBH

50 SEITEN
13 ABBILDUNGEN
5 TABELLEN

JUNI 1979

Inhalt

E i n f ü h r u n g

T. Martin

Im Rahmen des Aussprachetages "Prozeßrechner" der VDI/VDE-Gesellschaft Meß- und Regelungstechnik fand am 28. März 1979 in Dortmund ein Erfahrungsbericht mit Diskussion zur Prozeßprogrammiersprache PEARL statt. Als Beiträge waren vier ganz unterschiedliche Anwendungsbereiche ausgewählt worden:

- Steuerung der Netzleitstelle eines regionalen Elektroversorgungsunternehmens;

- Steuerung eines verfahrenstechnischen Chargenprozesses (Sudhaus einer Brauerei);

- Steuerung des Materialflusses in einem flexiblen Fertigungssystem (Modellanlage);

- Automatisierung eines industriellen Entwicklungslabors.

Dadurch bekamen die etwa 280 Teilnehmer einen Eindruck von dem breiten Anwendungsspektrum, das mit PEARL abgedeckt werden kann, sowie auch von drei verschiedenen PEARL-Rechnersystemen (von AEG, BBC und Siemens).
Als Hintergrund sei kurz der Stand der Bearbeitung von PEARL skizziert. Gegenwärtig gibt es etwa 140 PEARL-Rechneranwendungen, wovon die Mehrzahl bereits im industriellen Betrieb steht (Tabelle 1).

Anwendungsgebiet	Anzahl der Systeme
Metallverarbeitung, Walzwerke	37
Energieverteilung	31
Energieerzeugung	4
Grundstoffe, Chemie	16
Wasserversorgung	10
Andere Dienstleistungen (Fernsehen, Verkehr, Raumfahrt, etc.)	7
Andere industrielle Anwendungen	9
Versandhäuser, Lagerwesen	7
PEARL-Entwicklung und -Ausbildung	15

Tabelle 1:

Erfahrungen mit PEARL (Stand: Herbst 78)

(Quelle: Eigene Sammlung)

Die Anzahl der auf dem Markt verfügbaren PEARL-Rechner steigt langsam aber stetig (Tabelle 2), jedoch sind noch nicht alle Systeme mit hinreichend komfortabler Programmierumgebung ausgestattet. Erfreulich ist, daß auch ausländische Hersteller selbständig PEARL-Produkte auf den Markt bringen: Digital Equipment, MODCOMP und Norsk Data haben PEARL für 1980 angekündigt und sind damit auf der Hannover-Messe 1979 erstmals an die Öffentlichkeit getreten. Auch die ersten Versuche, leistungsfähigere Mikroprozessoren mit PEARL zu programmieren sind sehr erfolgreich verlaufen. Es handelt sich um folgende Aktivitäten:

- "Mehrprozessor-PEARL" des IITB, Fraunhofer Gesellschaft, Karlsruhe, das auf Siemens 310-kompatiblen, örtlich verteilten Rechnern läuft;

- PEARL für LSI11 von DEC von SEL, Stuttgart (zunächst für das Spacelab-Experiment verwendet);

- MUDAS-432-Datenprozessor von Dornier System, Friedrichshafen;

- PEARL auf Z80, Universitäten Karlsruhe und Erlangen (Anwendung im bedarfsgesteuerten Nahverkehr).

Firma	Rechnertyp
AEG-Telefunken	AEG 80-20
BBC	DP 1000/1500
Siemens	Siemens 330/340-Familie
Dietz (GPP)	Mincal 621
Krupp-Atlas	EPR 1100/1300/1500
MBP/Werum	HP 3000
MBP/Werum	Siemens 404/3
ESG	Litef Spirit 3
(einige auch als Cross-Compiler)	
Angekündigt:	
DIGITAL	DEC PDP11-Familie
Norsk Data (Werum)	Nord 10S
MODCOMP	
TU Berlin (Werum)	HP 21MX

Tabelle 2:

PEARL-Implementationen (Stand: 1978)

(Die Angaben in Klammern geben die Herkunft des Compilers an.)

In der auf die Erfahrungsberichte folgenden Abschlußdiskussion wurde keine Kritik an PEARL selbst laut, wohl aber an den Programmiersystemen, mit denen PEARL-Rechner derzeit ausgerüstet sind. Fragen wie das On-line-Nachladen von Programmen oder das Ankoppeln vorhandener Assembler- oder FORTRAN-Programme sind nicht überall gelöst.
Hier sind die Rechnerhersteller aufgerufen, Ihre PEARL-Systeme weiter zu verbessern.

Die Frage der Übertragbarkeit von PEARL-Programmen war in den Erfahrungsberichten nicht beantwortet worden. Hierzu berichtete T. Martin von einem Testprogrammpaket (entwickelt an der Universität Stuttgart), das von allen Rechnerherstellern zum Testen ihrer PEARL-Implementationen mit verwendet wird. Dadurch, daß sich ein großes PEARL-Programmpaket praktisch problemlos auf die unterschiedlichsten Rechner übertragen ließ, darf das Portabilitätskonzept von PEARL (Systemteil/Problemteil-Trennung) als verifiziert gelten.

Insgesamt muß festgestellt werden, daß die bisherigen erfolgreichen PEARL-Anwendungen noch zu wenig bekannt sind. Viele Anwender sind gegenüber PEARL aus Unkenntnis heraus noch skeptisch. Aus diesem Grund ist der PEARL-Verein, dessen Gründung gegenwärtig eingeleitet wird, so wichtig. Seine Hauptaufgabe sollte zunächst sein, die Anwendungserfahrungen mit PEARL auszuwerten und zu verbreiten.

Es folgen die vier auf dem Aussprachetag vorgetragenen Erfahrungsberichte.

U. M a y r , Regensburg

Basic PEARL, die Erfüllung einer Hoffnung?

Die Energieversorgung Ostbayern AG (OBAG) baut derzeit in Deggendorf die zweite von vier Netzleitstellen mit Doppelrechnersystem. Der überwiegende Teil der Anwenderprogramme für Deggendorf wird in PEARL geschrieben. Nachfolgend wird kurz die Problemstellung erläutert, danach die Gründe für die Anwendung von Basis-PEARL und schließlich ein Überblick über die derzeit vorliegenden Erfahrungen mit Basic PEARL.

- Das Projekt

  Die Energieversorgung Ostbayern AG (OBAG) in Regensburg ist ein regionales Elektrizitätsversorgungsunternehmen (EVU). Sie versorgt eine Fläche von 22 200 km^2 mit elektrischer Energie und somit ein Drittel von ganz Bayern. Bezogen auf die Fläche ist jedoch die abgegebene elektrische Arbeit relativ gering, da wenig Ballungsräume, dafür aber sehr flächige Gebiete versorgt werden müssen. Zur sicheren Führung des ausgedehnten 110- und 20-kV-Netzes und zur raschen Behebung von eventuellen Störungen sollen für das Gesamtgebiet der OBAG 4 Netzleitstellen gebaut werden, darüber hinaus wird später zur Koordinierung der Arbeiten in den 4 Netzleitstellen eine Netzleitung errichtet.

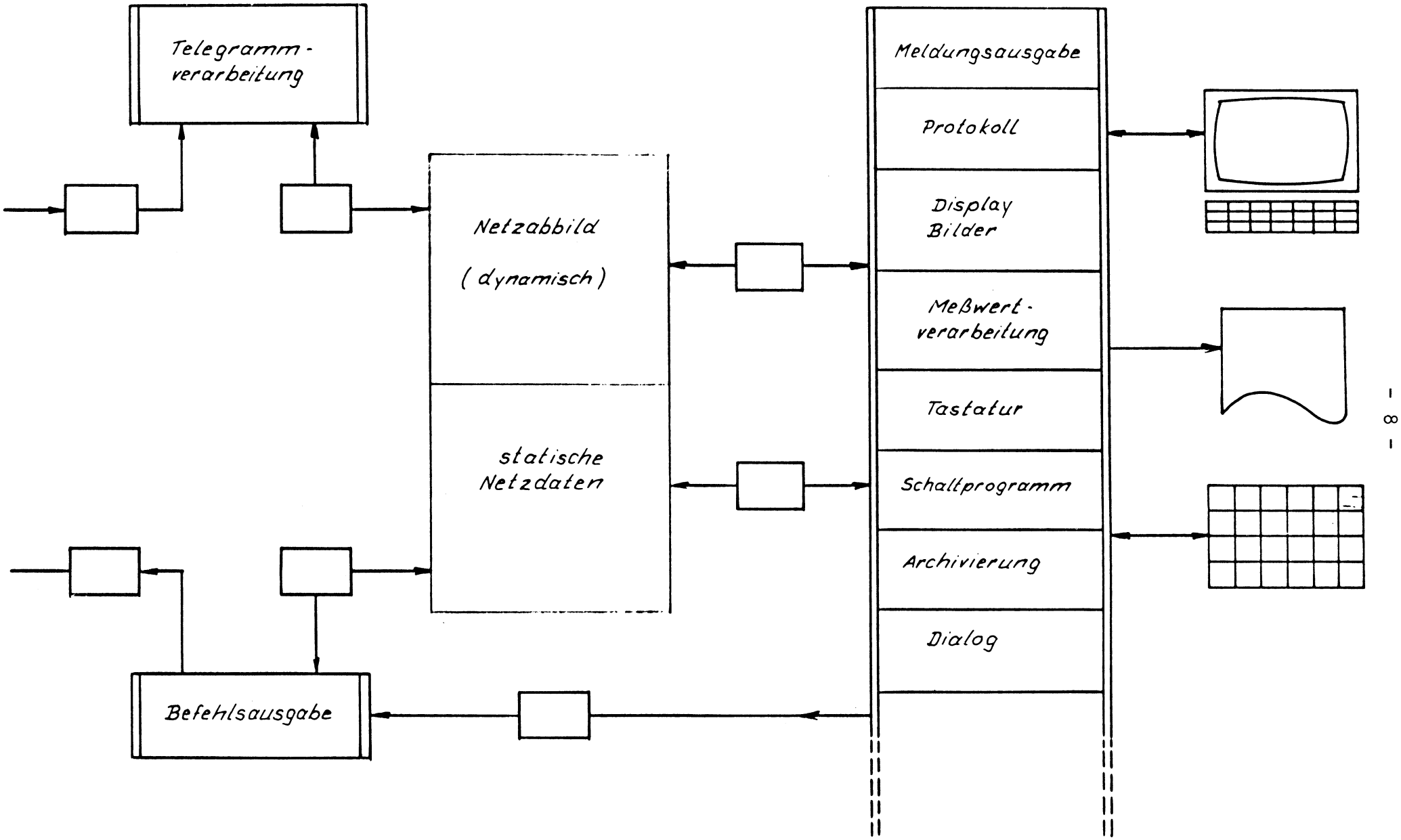
  Jede dieser 4 Netzleitstellen ist zuständig für das Verteilungsnetz in einem Gebiet von rd. 6000 km^2. In diesem Gebiet werden rd. vierzig Schalthäuser im 20-kV-Netz und dreißig 110/20-kV-Umspannwerke fernüberwacht, ferngemessen und ferngesteuert. Um den enormen Anfall an Informationen in einer Netzleitstelle zu bewältigen, wird jeweils ein Doppelrechnersystem eingesetzt. Alle Informationen werden über Farbsichtgeräte dargestellt und angezeigt. Die Befehle werden über eine funktionelle Tastatur eingegeben, die einzelnen Bedienschritte werden im Schaltbild auf dem Farbsichtgerät angezeigt. Darüber hinaus erledigt

das Rechnersystem zusätzliche Aufgaben, wie Protokollierung aller Betriebsabläufe, Überwachung von Meßwerten, Aufzeichnen von Meßwerten und zusätzliche für den Betrieb wichtige Überwachungs- und Steuervorgänge. (siehe Bild 1).

- Warum Basic PEARL ?

Das Programmsystem der ersten Netzleitstelle in Schwandorf ist vollständig in Maschinensprache (Assembler) geschrieben. Bei der Ausschreibung für die Netzleitstelle Deggendorf wurde die Erstellung des Programmsystems in der neuen Programmiersprache PEARL gefordert.Die Resonanz der angeschriebenen Unternehmen auf diese Forderung war seinerzeit (1975) unterschiedlich. Es wurden jedoch von allen renommierten Unternehmen Angebote abgeliefert, die diese Forderung berücksichtigten.
Die 4 Netzleitstellen der OBAG werden mit identischen Aufgaben erstellt. Um zu einer kontinuierlichen Auslastung des Personals zu kommen, kann jeweils nur eine Netzleitstelle realisiert werden. Die Zeit von der Planung bis zur Inbetriebnahme einer Netzleitstelle beträgt etwa 5 Jahre, so daß die Bauzeit für das gesamte System nach derzeitigen Abschätzungen rd. 20 Jahre betragen wird. Dies hat jedoch zur Folge, daß in jeder Netzleitstelle ein anderes Rechnersystem zur Aufstellung kommt, da die einzelnen Systeme bei der derzeitigen stürmischen Entwicklung der Elektronik bereits nach kurzer Zeit veraltet sind. Daraus ergibt sich jedoch wiederum, daß bei einer Programmerstellung in Assembler viermal die identischen Funktionen in vier unterschiedlichen Maschinensprachen realisiert werden müßten, d.h. das gesamte Programmsystem müßte abgesehen von der Funktionsanalyse, viermal völlig neu erstellt werden. Dies wäre jedoch aus wirtschaftlichen Gründen nicht tragbar.

Die Funktionen aller 4 Netzleitstellen sind, wie gesagt, identisch. Das bedeutet jedoch, daß im Laufe der Zeit in allen 4 Netzleitstellen entsprechend einer durch Erfahrungen abgeänderten Aufgabenstellung oder durch zusätzliche Wünsche



OBAG

Bild 1:
Softwaresystem Netzleitstelle Deggendorf

der Betriebsabteilungen neue Funktionen implementiert werden müssen. Das heißt jedoch im Falle einer Assembler-Programmierung, daß wiederum identische Funktionen viermal programmiert, implementiert und ausgetestet werden müßten.

Nur im Falle einer höheren Programmiersprache ist dieser unnötige Aufwand zu vermeiden, die Änderungen müssen nur einmal für alle 4 Netzleitstellen durchgeführt werden.

Wenn man nun über den Aufgabenbereich des eigenen Unternehmens hinausblickt, dann gewinnt das Thema der Mehrfachverwendung von Programmen eine wesentlich größere Bedeutung. Viele deutsche EVUs betreiben bereits Netzleitstellen, eine wesentlich größere Anzahl wird derartige Anlagen in den nächsten Jahren errichten. Obwohl die Mehrzahl der im Programmsystem einer Netzleitstelle realisierten Funktionen bei allen Unternehmen nahezu identisch ist, wurden bisher fast alle Systeme individuell programmiert. Dies ist volkswirtschaftlich nicht vertretbar. Die Elektrizitätsversorgungsunternehmen könnten bei Verwendung einer höheren Programmiersprache durch Austausch und Verwendung von Programmteilen Erhebliches einsparen.

Aus den obigen Ausführungen geht hervor, daß die Lösung dieser Probleme nur in einer höheren Programmiersprache resultieren kann. Nachdem man nach Fertigstellung der ersten Netzleitstelle zu dieser Erkenntnis gekommen war, wurden alle momentan auf dem Markt befindlichen Programmiersprachen auf ihre Eignung für Prozeßrechneranwendungen hin untersucht. Das Ergebnis war bei den bisher bekannten Programmiersprachen entmutigend (FORTRAN, PL/1, ALGOL usw.).

Als einzige Sprache besitzt PEARL echte Sprachmittel für die Koordinierung und Aktivierung von Programmteilen sowie für prozeßrechnerspezifische Manipulationen.

Obgleich es zur damaligen Zeit keine großen Anwendungen gab, wagte man mit der Festlegung auf diese Programmiersprache den Sprung in die Zukunft.

- Projektabwicklung

Zu Beginn der Arbeiten wurde, aufbauend auf der bereits von einer vorhergehenden Netzleitstelle vorliegenden Funktionsanalyse, der gesamte Problemkomplex in einzelne Teile zerlegt. Dabei wurde dann auch festgelegt, daß einige Teile des Programmsystems, die besonders Hardware-nah sind, noch in Assembler geschrieben werden sollten. Hierbei handelt es sich um die Programmteile, die die einlaufenden Fernwirktelegramme übernehmen und im Prozeß-abbild ablegen. Dies geschah deshalb, weil zum damaligen Zeitpunkt über die Auswirkungen von PEARL auf das Zeit-verhalten noch nichts bekannt war. Man wollte bei diesen relativ zeitkritischen Problemen jegliche Gefahr aus-schließen.

Nachdem alle Funktionen nochmals genauestens durchge-arbeitet und beschrieben waren, wurden die einzelnen Teil-probleme einer exakten Systemanalyse unterzogen.

Die vorgenannten Arbeiten haben mit der verwendeten Programmiersprache nichts zu tun. Es ist vielmehr so, daß -unabhängig von der Sprache- Funktions- und System-analyse sehr sorgfältig durchgeführt werden müssen. Eventuelle Versäumnisse in diesen beiden Punkten führen später mit Sicherheit zu wesentlich größeren Schwierig-keiten.

Bei der nachfolgenden Codierung zeigten sich erstmals die wesentlichen Vorteile von PEARL. Neben den von anderen höheren Programmiersprachen bekannten Vorteilen (geringer Schreibaufwand, leichte Lesbarkeit sowie Selbstdokumentation) zeigt sich hier vor allem der Nutzen der in PEARL erstmalig verfügbaren Sprachmittel zur Einplanung und Koordinierung von parallel ablaufenden Programmen.

Das Programmsystem für eine Netzleitstelle ist sehr umfang-reich und komplex. Im vorliegenden Fall besteht es aus rd. 50 eigenständigen Programm-Modulen (Task), die einzeln angesprochen und aktiviert werden können.

Diese einzelnen Module müssen nun im Gesamtsystem verankert werden, sie müssen zyklisch oder spontan angestoßen werden, darüber hinaus ist eine gegenseitige Koordinierung erforderlich. Für all diese Funktionen bietet PEARL echte Sprachmittel, mit denen diese recht komplexen Funktionen in einfacher Weise niedergeschrieben werden können. All diese Anweisungen sind, weil sie einer höheren Sprache entspringen, sehr transparent und darüber hinaus selbstdokumentierend.

Bei der nachfolgenden Implementation und dem Test der einzelnen Programmteile erweist sich das vorstehend Erwähnte als weiterer Vorteil. Dadurch, daß das Programm sehr leicht lesbar ist, sind Fehler wesentlich leichter aufzufinden und zu korrigieren. Außerdem ist dann mit der Programmliste bereits ein Teil der Dokumentation erledigt.

Die Arbeiten für das Programmsystem sind derzeit im Gang. Bisher ist ein Teil der Programm-Module fertiggestellt. Das Gesamtsystem wird etwa Anfang 1980 in Betrieb genommen. Nichtsdestoweniger kann man heute schon auf einige Erfahrungen mit der neuen Prozeßrechner-Programmiersprache PEARL zurücksehen. Alle im Software-Team beteiligten Personen hatten vorher noch nicht mit PEARL gearbeitet, sie hatten bislang Projekte in Assembler erstellt. Somit war eine anfängliche Skepsis gegenüber der neuen Sprache durchaus verständlich; im nachhinein stellt sich heraus, daß diese anfänglich kritische Betrachtungsweise dem Ganzen durchaus nützlich war. Heute jedoch sind alle Beteiligten von den Vorteilen der Programmiersprache PEARL überzeugt und arbeiten gern damit.

- Zusammenfassung und Ausblick

Allen Beteiligten war von Anfang an klar, daß mit Basic PEARL noch kein Optimum geschaffen werden konnte. Der hier definierte Sprachumfang entsprach dem damaligen Wissens- und Erkenntnisstand der Beteiligten.Er war andererseits aber auch ein gewisser Kompromiß, um diesen gemeinsamen Sprachumfang möglichst schnell implementieren und in Form von Compilern auf den Markt bringen zu können.

Darin lag ein gewisses Risiko, doch war dies nicht zu vermeiden, da nur in der praktischen Erprobung in größeren Pilotprojekten der Beweis für die Verwendbarkeit der Sprache erbracht werden kann. Andererseits war man sich darüber im klaren, daß eine weitere Verbesserung oder Ergänzung des Sprachumfanges nur auf Grund praktischer Erfahrungen in umfangreichen und komplexen Software-Systemen möglich ist.

Nach nunmehr fast 1-jähriger Arbeit mit der Programmiersprache PEARL kann man sagen, daß PEARL ein hervorragendes Mittel zur Lösung auch sehr komplexer Prozeßprobleme ist. Es ist bislang kein Teilproblem aus der Netzleitstellen-Technik bekannt, das nicht mit dem vorliegenden Sprachumfang von Basic PEARL gelöst werden könnte. Es hat sich aber gezeigt, daß an manchen Stellen noch einige Verbesserungen anzubringen sind. Es handelt sich hierbei nicht um Fehler oder um fehlende Eigenschaften im Grundkonzept von Basic PEARL, sondern um kleine Dinge, die für die Praxis geändert oder verbessert werden müssen, um die Programmierarbeit einfacher und eleganter durchführen zu können.
Wenn man die bisherigen Erfahrungen in einem Satz zusammenfassen will, dann kann man sagen, daß es sich lohnt, an Basic PEARL weiterzuarbeiten und diese Sprache so abzurunden, daß sie ein handliches Werkzeug für den Praktiker wird.

- Wünschenswerte Änderungen und Erweiterungen

Nachfolgend sollen einige Punkte aufgeführt werden, die in Basic PEARL ergänzt oder zugefügt werden sollen. Der Platz erlaubt es nicht, hier jeden einzelnen Punkt in aller Ausführlichkeit darzulegen, die Wünsche und Ergänzungen sollen nur kurz angerissen werden. Die Aufzählung erhebt auch keinen Anspruch auf Vollständigkeit

  - die Anforderung und Freigabe von Semaphor-Variablen (REQUEST, RELEASE) soll zeitmodifiziert möglich sein (AFTER, AT), dadurch können zusätzliche Tasks eingespart werden

- es fehlen Standard-Operatoren, die einem Programm die aktuelle Uhrzeit sowie das Datum liefern. Dies ist für die Prozeßdaten-verarbeitung unabdingbar

- es fehlt die Möglichkeit, beim Start einer Task Daten zu übergeben, diese elegante Möglichkeit würde erheblichen Zusatzaufwand einsparen

- entsprechend dem Operator ".BIT" ist der äquivalente Operator "CHAR" erforderlich, darüber hinaus muß es möglich sein, Teilbereiche aus Bit- und Zeichenketten zu selektieren, d.h. also, es werden erforderlich die Operatoren ".BIT (m-n)" und ".CHAR (m-n)".

Darüber hinaus ist es notwendig, einheitliche Richtlinien für das Umfeld der Sprache festzulegen, d.h. für das Test- und Bediensystem. Dieser Bereich wurde bislang den Implementatoren überlassen mit dem Erfolg, daß es hier zu enormen Unterschieden kam. Es ist jedoch sinnlos, die Sprache zu normen, wenn man doch wieder jeweils einen Spezialisten braucht, der das System zum Laufen bringt.
Wie bereits erwähnt, handelt es sich bei den vorgenannten Punkten um Dinge, die dem Programmierer das Leben leichter und die Sprache PEARL handlicher machen.
Nachdem von Seiten der Bundesregierung erhebliche Mittel in die Förderung von PEARL investiert wurden, besteht die berechtigte Hoffnung, daß über PDV auch diese Aktivitäten zur Abrundung von PEARL noch gefördert und koordiniert werden können, damit PEARL wirklich abgeschlossen werden kann.

Praktischer Einsatz von PEARL am Beispiel der Brauereiautomatisierung

H. Eggert
HENNINGER-BRÄU KGaA
Frankfurt am Main

1. Einleitung

Im Rahmen des vom Kernforschungszentrum Karlsruhe geförderten Forschungsauftrages "Brauereiautomatisierung mit Prozeßdatenverarbeirungsanlagen" wurde ein Teil einer Brauerei, der Anlagenbereich eines Sudhauses /1/ unter Verwendung der Echtzeitprogrammiersprache PEARL automatisiert.
Das Ergebnis jeder Programmierung sollte ein Programm sein, das sich selbst dokumentiert und damit eine möglichst genaue Abbildung des Lösungsweges darstellt.
Dazu werden hier vorzugsweise die Objekte des Lösungsweges, d. h. partiell die Prozeßdaten und Prozeßlenkungsalgorithmen hervorgehoben. Die Elemente der Programmiersprache werden lediglich als Beschreibungswerkzeug benutzt, ohne sie selbst, d. h. ihre Syntax und Semantik vollständig zu beschreiben.
Bezüglich des strukturellen Aufbaus sowie der Sprachelemente von PEARL wird auf die entsprechende Literatur verwiesen /2, 3, 4/.

2. Der Brauprozeß

In dem Anlagenbereich des Sudhauses läuft verfahrenstechnisch im Chargenbetrieb der eigentliche Brauprozeß (Sudprozeß) ab. Dabei wird aus den Rohstoffen Malz (gekeimte Gerste), Wasser und Hopfen die sogenannte Würze bereitet, welche in einem weiteren Anlagenbereich, dem Gärkeller, unter Zugabe von Hefe zum fertigen Bier vergoren wird.
Älteste Darstellungen beweisen, daß der Brauprozeß schon seit Jahrtausenden bekannt ist, wenn er auch früher mit anderen Mitteln als heute durchgeführt wurde.
Die folgende Darstellung /5/ zeigt einen Teil des Sudprozesses, wie er bei den "alten Ägyptern" ablief:

Abb. 1: Ägyptische Wandmalerei aus dem Grab des Kenamon in Luxor, etwa 1500 v. Chr.

Brotfladen werden von Arbeitern gevierteilt und aus ihnen, zusammen mit Wasser, in Gefäßen die Maische, ein Vorprodukt der Würze, bereitet. Heute verfügen Brauereien dagegen über ein hohes, wissenschaftlich gestütztes Verfahrens-know-how sowie über eine moderne Anlagentechnik.

Das folgende Rohrleitungs- und Instrumenten-Fließbild /6 / zeigt einen Teil des Sudhausanlagenbereiches:

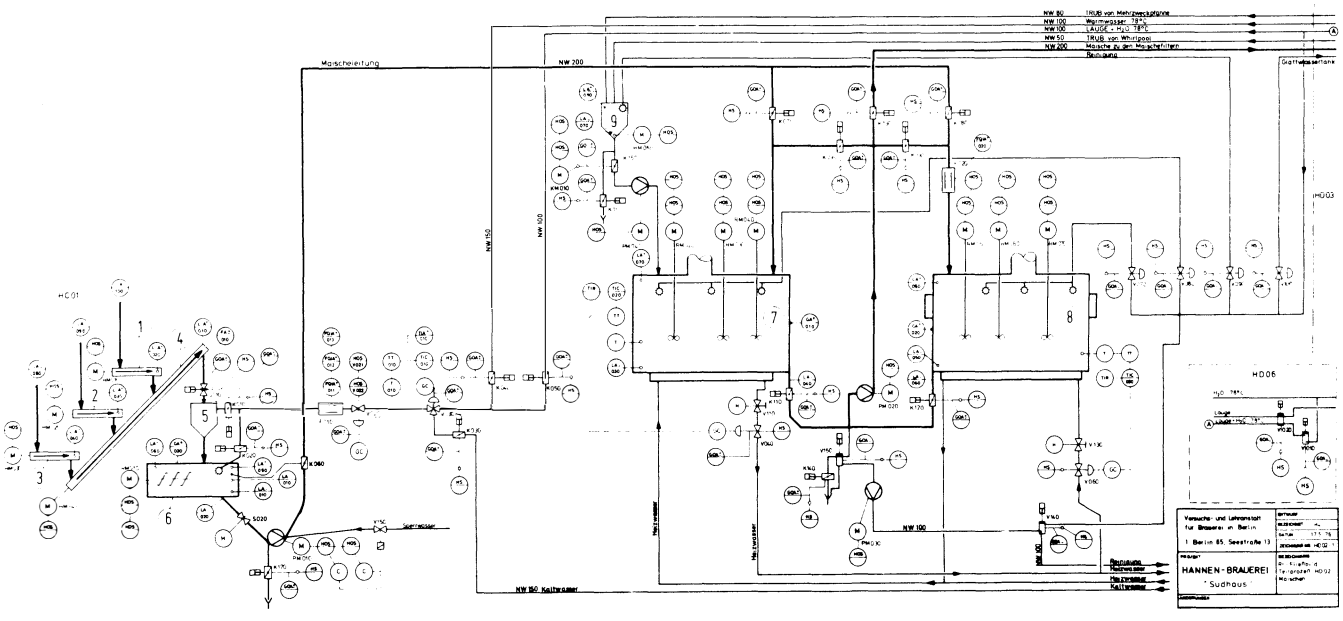


Abb. 2: Rohrleitungs- und Instrumenten-Fließbild

Im Prinzip läuft in diesem dargestellten Teilbereich der Anlage der gleiche Prozeß ab, wie ihn Abb. 1 veranschaulicht. Das gemahlene Malz (Schrot) gelangt über Fördersysteme (1, 2, 3, 4) in einen Vormaischer (5), wo eine Vermischung mit Wasser erfolgt und von dort in einen Einmaischtank (6). Von hier wird die Mischung (Maische) kontinuierlich in die Maischgefäße (7, 8) gepumpt (eingemaischt), in denen der eigentliche Maischprozeß abläuft.
Ohne auf die Einzelheiten einzugehen, läßt die Vielzahl der dargestellten Meß- und Stellglieder einen relativ komplizierten Prozeßablauf vermuten.

3. Das Hardwarekonzept

Das Hardwarekonzept ist gekenntzeichnet durch eine funktionale Trennung von Prozeßrechenanlage und Steuerwarte (für Handbetrieb) sowie durch eine strenge Gliederung der Hardwarekomponenten in organisatorischen Ebenen.

Die "Ebene der unterlagerten Steuerung" sorgt für eine Signalverstärkung. Außerdem wird hier die Steuerungslogik für die funktionale Trennung von Prozeßrechenanlage und Steuerwarte sowie die Verriegelungslogik als Sicherungsmaßnahme bei relevantem Handbetrieb bereitgestellt.

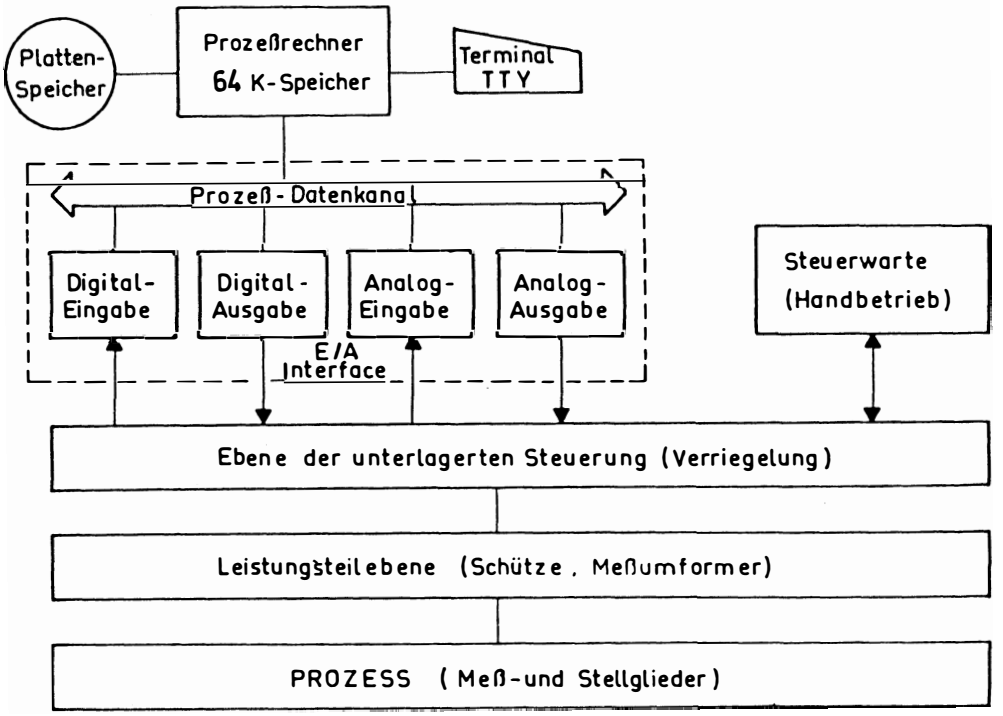


Abb. 3: Die Hardwarestruktur

4. Das Softwarekonzept

Das Softwarekonzept ist gekennzeichnet durch ein in PEARL erstelltes hierarchisches Anwendersystem. Bei dem Anwendersystem wird zwischen einer "statischen Programmorganisation" und einer "dynamischen Programmorganisation" unterschieden.

Die "statische Programmorganisation" betrifft die Handhabbarkeit bei der Übersetzungsphase. PEARL erlaubt eine Programmorganisation nach Modulen und zwar derartig, daß eine Änderung in einem Modul keine Änderung in einem anderen Modul erzwingt. Die Module können getrennt übersetzt werden, nur der Link- und Ladevorgang erfolgt gemeinsam.

Die Zuordnung des Lösungsweges der Automatisierungsaufgabe erfolgt in der Weise, daß im Hauptmodul, neben der Gerätebeschreibung im Systemteil, im Problemteil alle von der eigentlichen Prozeßbeschreibung losgelösten organisatorischen Aufgaben (z. B. Ein/Ausgaben, Interruptbearbeitung) beschrieben werden.

Die prozeßspezifischen Aufgaben (Prozeßlenkungsalgorithmen) werden, aus einer Teilprozeßorganisation abgeleitet, in einzelnen PEARL- Submodulen beschrieben.

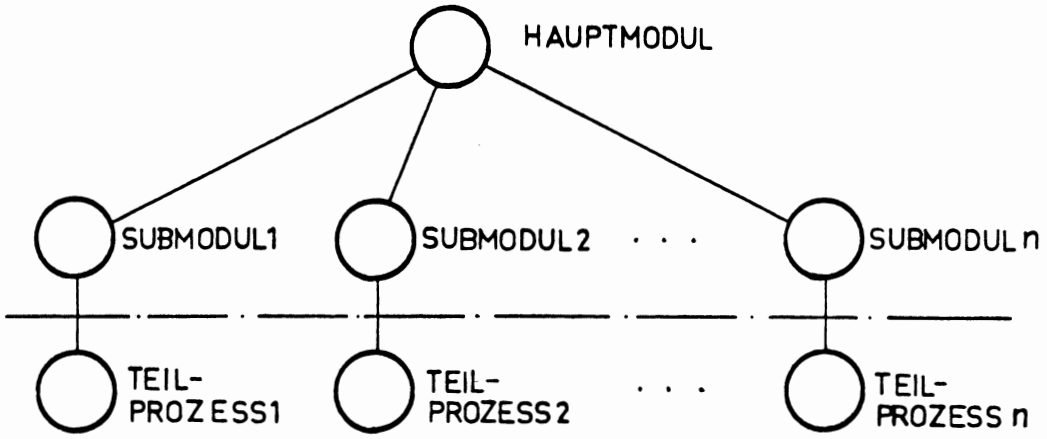


Abb. 4: Modulare Aufteilung von Prozeß und PEARL-Programm

Die "dynamische Programmorganisation" genügt den Merkmalen dynamischer Prozesse, d. h. die Beschreibung parallel laufender Prozesse der Realität erfolgt durch Programme mit dem Attribut "Task" unter Angabe einer Priorität.

Die Anwendersoftware erfüllt insgesamt die folgenden Aufgaben:

- Steuerung und Überwachung des gesamten Sudprozesses unter Berücksichtigung von Parametern, welche verschiedene Verfahrensmethoden sowie die Herstellung der Würze für vier Biersorten über einen Produktionszeitraum von einer Woche ermöglichen,

- Dialog mit dem Bediener am Wochenanfang, während des Verfahrensablaufes zur Synchronisation von Prozeßfunktionen (wenn on-line-Messungen nicht möglich sind) und im Störungsfall zur Entscheidung über die mögliche "Handfreigabe" für Teilprozesse,

- Routine- und Störungsmeldungen,

- Kontinuierliche Überwachung aller Stellglieder.

5. PEARL und die notwendige Abstraktionsebene zur Formulierung des Lösungsweges

Die Programmiersprache PEARL kann als "abstrakte Maschine", hinter der sich Computer, Prozeß und Mensch-Maschine-Kommunikation verbergen, betrachtet werden, welche Ausdrücke dieser Sprache verstehen kann.

Die Elemente von PEARL verfügen also über einen gewissen Grad an Abstraktion über die von der realen Maschine benutzten Objekte.

Zur Formulierung eines Lösungsweges ist es unklug, eine streng maschinenorientierte Programmiersprache anzuwenden, wie es nutzlos ist, Computerprogramme in einer derartig abstrakten Notation zu beschreiben, die sämtliche Darstellungsprobleme vernachlässigt.

Die notwendige Abstraktionsebene zur Problemformulierung wird in Abhängigkeit vom Anwendungsfall stets irgendwo zwischen den aufgezeigten Extremen liegen oder genauer ausgedrückt - die schrittweise Verfeinerung eines Algorithmus endet auf einer Abstraktionsebene, die für den Programmierer als "primitiv" bezeichnet werden kann.

Bei dem vorliegenden Anwendungsfall gilt als "primitiv", die dem Verfahrensingenieur und nicht die dem Computerspezialisten vertraute Terminologie.
Nun wäre es für jeden Anwender wünschenswert, eine Programmiersprache zu haben, deren Sprachelemente sich genau auf der für sein Problem gewünschten Abstraktionsebene befinden.
Das ist praktisch aber nicht direkt möglich, da dieser Wunsch im Widerspruch zu der notwendigen Forderung einer möglichst großen Allgemeingültigkeit einer Programmiersprache steht.
Die Programmiersprache PEARL bietet hier einen guten Kompromiß.
Einerseits verfügt PEARL über eine leicht überschaubare Menge allgemeingültiger Sprachelemente, die für jede Problemformulierung, also auch für die Beschreibung paralleler Prozesse gültig sind, wobei die strenge Typenbindung der Daten wesentlich zur Sicherheit der Programme beiträgt, da Fehler durch falsche Datenzuordnung schon zur Übersetzungszeit erkannt werden.
Andererseits bietet PEARL u. a. durch das Prozedurkonzept einfache Möglichkeiten,eine auf das spezielle Problem zugeschnittene Abstraktionsebene einzuführen und zu benutzen.

Bei dem vorliegenden Anwendungsfall werden die Sprachelemente von PEARL zur Beschreibung paralleler Prozesse sowie die Kontrollstrukturen direkt übernommen. Die Problemorientierung wird dabei durch frei wählbare Variablennamen unterstützt.

Alle Ein/Ausgabe-Mechanismen werden in Prozeduren "versteckt" und sind damit dem Anwender rein problembezogen zugänglich.

Bei dem abschließenden Beispiel aus der Anwendungsprogrammierung wird die folgende Teilmenge von PEARL-Sprachelementen benutzt:

a) Behandlung paralleler Prozesse und deren Synchronisation

- Operationen zur Aktivierung, Suspendierung und Fortsetzung von Tasks auf dem Datentyp TASK:

```
ACTIVATE (taskname);
SUSPEND  (taskname);
CONTINUE (taskname);
```

- Operation als Taskvorsatz beim Eintreffen eines Interrupts auf dem Datentyp INTERRUPT:

```
WHEN interrupt-name ...
```

- Operation als Taskvorsatz zur Wiederholung auf dem Datentyp DURATION:

```
EVERY duration-ausdruck ...
```

- Operation zur Verzögerung einer Task auf dem Datentyp DURATION:

```
AFTER duration-ausdruck RESUME;
```

- Operation zur Synchronisation von Tasks auf den Datentypen EVENT und SEMA:

```
SET      (ereignis);
RESET    (ereignis);
WAIT     (ereignis-ausdruck);
REQUEST  (semavariable);
RELEASE  (semavariable);
```

b) Kontrollstrukturen

```
IF bedingung
THEN DO;
          /* offener Abschnitt */
END;
ELSE DO;
          /* offener Abschnitt */
END;
```

c) Ein/Ausgabe

Operation zum Prozeduraufruf auf dem Datentyp ENTRY unter Benutzung von Parametern mit den Datentypen BIT und BINARY FIXED.

```
CALL stellgliedmanipulation(stellgliedname);
CALL sollwertausgabe(regelstrecke, sollwert);
CALL istwerteingabe(regelstrecke);
CALL meldung(textindex);
```

6. Anwendungsbeispiel

Auf die Beschreibung der "statischen Programmorganisation" (Hauptmodul, Submodul, Systemteil, Problemteil, Deklarationen) wird bewußt verzichtet, um die "dynamische Programmorganisation", die Behandlung paralleler Prozesse und deren Synchronisation, besonders hervorzuheben.

Das Anwendungsbeispiel orientiert sich an für Verfahrensprozesse geltenden Grundeigenschaften, wobei sich Chargen nach einem bestimmten Schema durch die Anlage bewegen.
Eine Charge wird angefahren, bearbeitet und abgefahren.

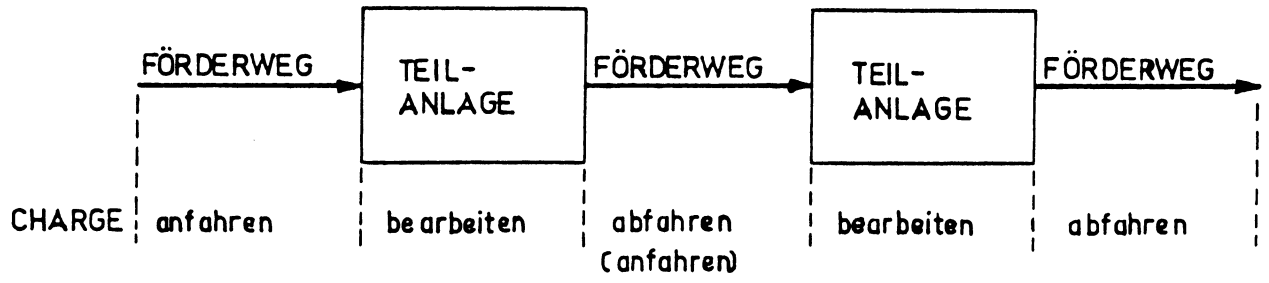


Abb. 5: Chargenfunktionen

Die Grundlage des Anwendungsbeispiels ist ein stark "abgemagertes" Rohrleitungs- und Instrumenten-Fließbild (s. Abb. 6).

Die Prozeßbeschreibung entspricht nicht ganz der Realität.
Es mußte jedoch ein Kompromiß gefunden werden, da der Prozeß hier einerseits nicht annähernd vollständig beschrieben werden kann, jedoch andererseits die Prozeßbeschreibung nichttriviale Merkmale der

Parallelität enthalten soll, um die dazu in PEARL vorhandenen Möglichkeiten aufzeigen zu können.

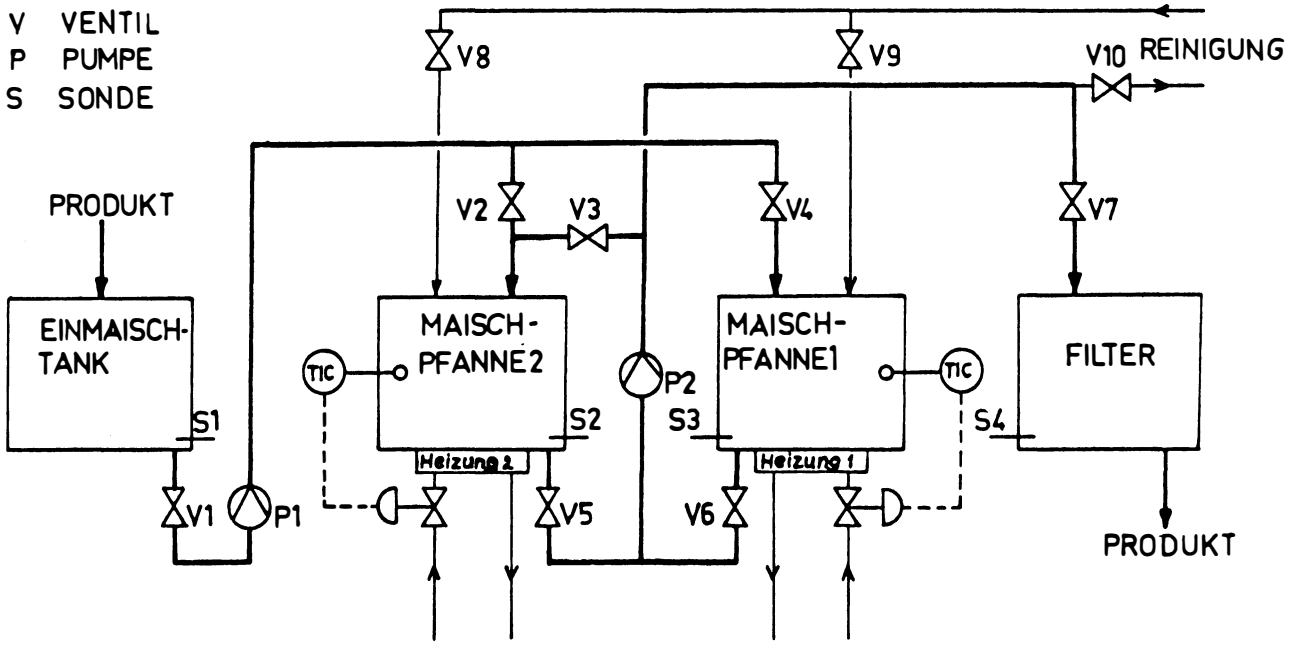


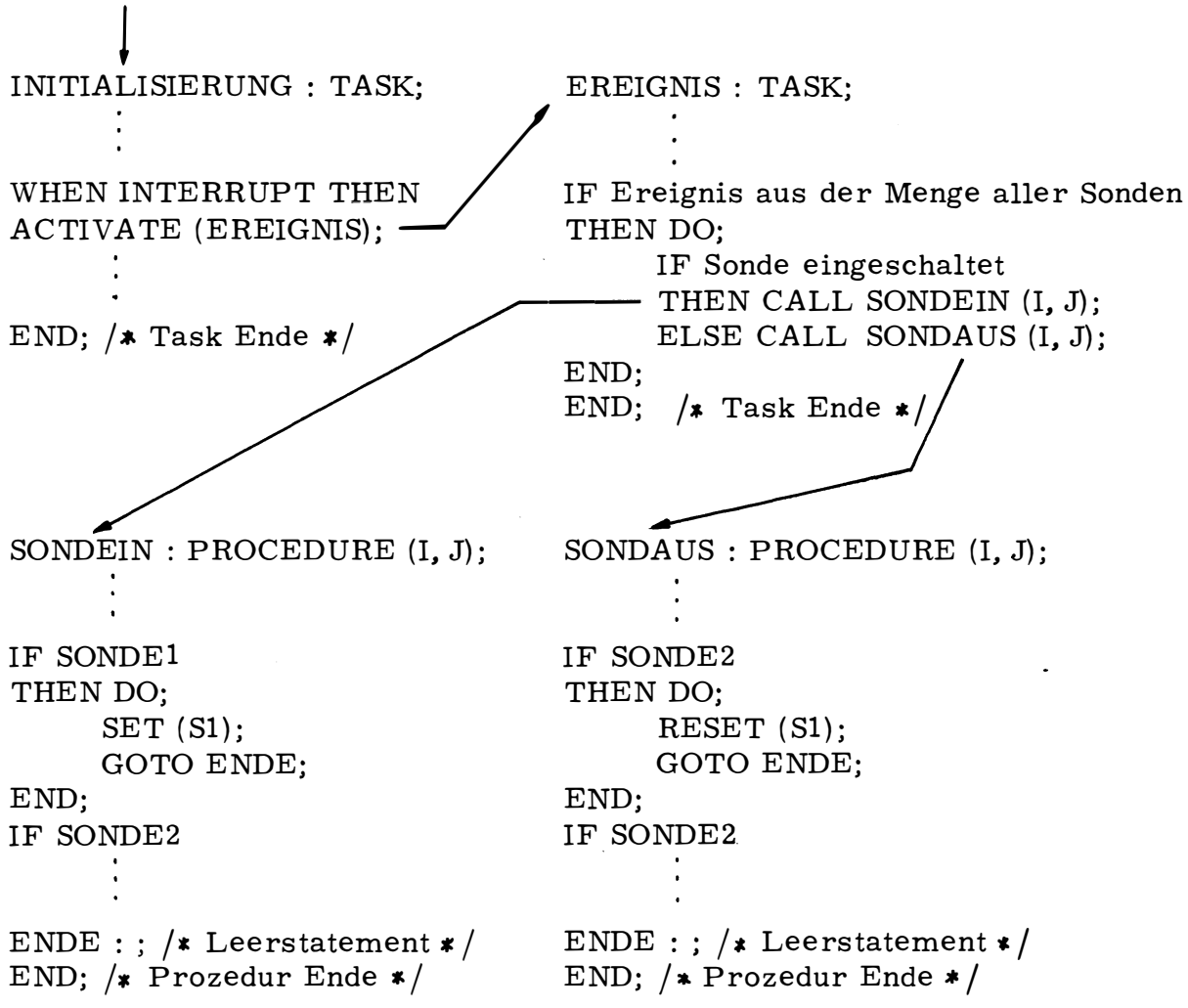
Abb. 6: Rohrleitungs- und Instrumenten-Fließbild für das Anwendungsbeispiel

Die Charge gelangt vom Einmaischtank bei einem bestimmten Verfahren zum Teil in die Maischpfanne 1 und zum Teil in die Maischpfanne 2. In diesen Behältern wird die Charge durch dosierte Erwärmung und andere Verfahren bearbeitet.
In der Maischpfanne 2 wird die Charge schließlich wieder zusammengeführt und weiterbearbeitet. Nach Beendigung der Bearbeitung gelangt die Charge zum Filter.

Die Heizflächen der Maischbehälter verkleben mit der Zeit, so daß die Heizleistungen unterschritten werden. In Abhängigkeit von den Heizleistungen kann eine Reinigung der Maischbehälter eingeleitet werden. Während der Reinigung darf keine neue Charge angefahren werden. Eine Synchronisierung der parallel laufenden Prozesse erfolgt u. a. durch ereignisauslösende Füllstandssonden.

Die Mitteilung von Prozeßereignissen an die Prozeßlenkungsprogramme bzw. das PEARL-Betriebssystem erfolgt durch die folgenden Programme:

Aktivierung von der Bedientask



Die Prozeßlenkungsalgorithmen werden durch die folgenden Programme beschrieben:

Aktivierung von einer anderen Prozeßlenkungstask

```
EINMAISCHEN : TASK;
        :
IF MAISCHVERFAHREN = A
THEN DO;
     WAIT (¬S2 & ¬S3);
     /* Charge abfahren, anfahren */
     CALL AUF (V1);
     CALL AUF (V4);
     CALL EIN (P1);
     ACTIVATE (MAISCHEN1);
     AFTER 8 MIN RESUME;
     CALL AUF (V2);
     CALL ZU (V4);
     ACTIVATE (MAISCHEN2);
     WAIT (¬S1);
     CALL AUS (P1);
     CALL ZU (V1);
     CALL ZU (V2);
END;
IF MAISCHVERFAHREN = B
        :
END; /* Task Ende */
```

```
MAISCHEN1 : TASK;
        :
REQUEST (MAISCHPFANNE1);
WAIT (S3);
/* Charge bearbeiten */
        :
CALL SOLL (HEIZUNG1, SOLLWERT);
EVERY 1 MIN ACTIVATE (UEBWA);
AFTER 15 MIN RESUME;
PREVENT (UEBWA);
        :
        :
/* Charge abfahren */
CALL AUF (V6);
CALL AUF (V3);
CALL EIN (P2);
CONTINUE (MAISCHEN2);
WAIT (¬S3);
CALL AUS (P2);
CALL ZU (V6);
CALL ZU (V3);
RELEASE (MAISCHPFANNE1);
END; /* Task Ende */
```

s. ums.

```
MAISCHEN2 : TASK;
        :
REQUEST (MAISCHPFANNE2);
WAIT (S2);
/* Charge bearbeiten */
        :
SUSPEND;
/* Charge weiterbearbeiten */
        :
        :
/* Charge abfahren */
WAIT (¬S4);
CALL AUF (V7);
CALL AUF (V5);
CALL EIN (P2);
WAIT (¬S2);
CALL AUS (P2);
CALL ZU (V5);
CALL ZU (V7);
RELEASE (MAISCHPFANNE2);
END; /* Task Ende */
```

Aktivierung s. S. vorher

```
UEBWA : TASK;
        :
CALL IST (HEIZUNG1);
IF ISTWERT < SOLLWERT - 1
THEN DO;
     CALL MELDUNG (I);
     ACTIVATE (REINIGUNG);
     PREVENT;
END;
IF ISTWERT > SOLLWERT + 1
THEN DO;
     I : = I + 1;
     CALL MELDUNG (I);
END;
END;  /* Task Ende */
```

```
REINIGUNG : TASK;
        :
REQUEST (MAISCHPFANNE1);
CALL AUF (V9);
AFTER 10 MIN RESUME;
CALL AUF (V10);
CALL AUF (V6);
CALL EIN (P2);
WAIT (¬ S3);
CALL AUS (P2);
CALL ZU (V6);
CALL ZU (V10);
/* Nachspülen */
        :
RELEASE (MAISCHPFANNE1);
END;  /* Task Ende */
```

7. Literaturhinweise

/1/ Narziss, L.:
Abriß der Bierbrauerei
Ferdinand Enke Verlag Stuttgart, 1972

/2/ Basic PEARL, Language Description
KfK - PDV 120, 1977

/3/ Full PEARL, Language Description
KfK - PDV 130, 1977

/4/ Kappatsch, A.:
Überblick über die Echtzeitprogrammiersprache PEARL
KfK - PDV 140, 1977

/5/ Jung, H.:
Bier-Kunst und Brauchtum
Schropp Verlag Dortmund

/6/ DIN 28004: Fließbilder verfahrenstechnischer Anlagen

DIN 2429: Sinnbilder für Rohrleitungsanlagen

DIN 19227: Bildzeichen und Kennbuchstaben für
Messen, Steuern, Regeln in der Verfahrenstechnik

EIGNUNG DER SPRACHMITTEL VON PEARL ZUR PROGRAMMIERUNG DER MATERIALFLUSSSTEUERUNG IN FLEXIBLEN FERTIGUNGSSYSTEMEN

J. Kremser, Dortmund

1. Einleitung - Kurzbeschreibung des flexiblen Fertigungssystems

1.1 Beschreibung der Modell Hardware und der Arbeitsweise

Die Programmiersprache PEARL wurde bereits an einigen deutschen Prozeßrechnertypen implementiert und wird den Anwendern zur Verfügung gestellt. Sie erweitert die Wahlmöglichkeiten des Anwenders auf dem Gebiet der Echtzeit-Programmiersprachen.

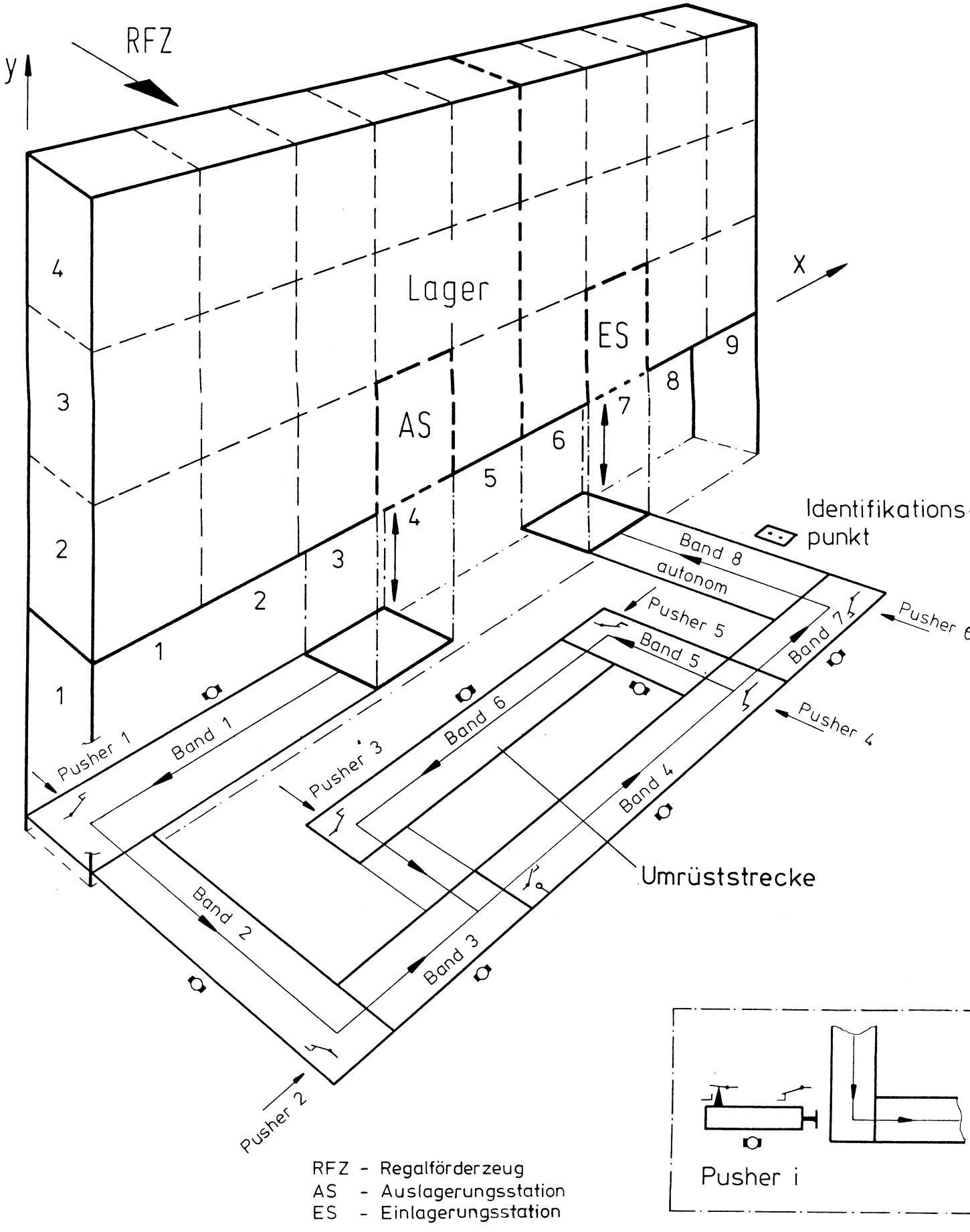
Auf dem fertigungstechnischen Gebiet sind in der nächsten Zeit ebenfalls Neuerungen zu erwarten. Damit sind vor allem sog. flexible Fertigungssysteme gemeint, die in ihrem Kern an einer neuen Materialflußkonzeption in der Fertigung basieren.

Im folgenden wird über den Einsatz von PEARL zur Programmierung des Modells eines flexiblen Fertigungssystems berichtet. Der zur Verfügung gestellte PEARL-Subset ging weit über den Umfang des Basis-PEARL-Subsets hinaus. Die Bearbeitung der Aufgabe hat einige typische Problemstellungen aufgezeigt, die charakteristisch für Förder- und Lagerprozesse sind.

Die Software sollte folgende Anlagenteile des Modells kontrollieren:

1) Rüstbereich
2) Regalförderzeug
3) Lager
4) Fertigung

Der Rüstbereich umfaßt ein aus Rollenbahnen und Pushern bestehendes Fördersystem (Bild 1). Die Pusher befördern die Werkstücke an Stellen, an denen die Förderrichtung auf der Rollenbahn um 90° geändert werden muß, um den nächsten Rollenbahnabschnitt zu erreichen. Die Werkstücke können im Rüstbereich von den Rollenbahnen entnommen werden bzw. nach Durchlauf der Umrüststrecke der Einlagerungsstation zugeführt werden. Zur Kontrolle der Pusher und der Fortbewegung der Werkstücke wurden an den Rollenbahnen Endschalter eingesetzt. Die Schalter sind an die Interruptanschlüsse (Prozeßunterbrechungseinheiten) des Rechners geführt. Zum Rüstbereich gehört ein Identifikationspunkt, an dem die Werkstückcodierung abgefragt wird. Entsprechend dieser Codierung wird die Fertigungsreihenfolge der Werkstücke vom Rechner festgelegt. Die fördertechnischen Schnittstellen



RFZ - Regalförderzeug
AS - Auslagerungsstation
ES - Einlagerungsstation

Bild 1: Rüstbereich und Lager

zwischen Rüstbereich und Regalförderzeug (RFZ) bilden die Einlagerungs- und Auslagerungsstationen (ES und AS), die durch Hub- bzw. Senkeinrichtungen erreicht werden. Die Motoren der Rollenbahnen sowie der Ein- und Auslagerungsstationen wurden über Relais an den Rechner angeschlossen.

Das Lager beinhaltet 25 Werkstückplätze, sowie die Einlagerungs- und Auslagerungsstationen. Es wurde hauptsächlich in der Funktion eines Pufferlagers für die Bearbeitungsstationen benutzt. Im Lager werden diejenigen Werkstücke zwischengelagert (gepuffert), deren als nächste vorgesehene Bearbeitungsstation besetzt ist. Die an der Einlagerungsstation (ES) aus dem Rüstbereich ankommenden Werkstücke werden vom Regalförderzeug übernommen. Ein Rechnerprogramm (Dispositionsmodul) entscheidet, ob das Werkstück ins Lager oder zu den Drehtischen der Bearbeitungsstationen gebracht werden soll. Der Entscheidung entsprechend erhält der Steuerungsmodul des Regalförderzeugs (RFZ) die Soll-Koordinaten, um diesen Transportauftrag erfüllen zu können. Die Daten werden aus einem Auftragspuffer entnommen.

Innerhalb des Fertigungsbereiches (Bild 2) sind 5 Bearbeitungsstationen (BS) vorhanden. Zu jeder Bearbeitungsstation gehört ein BS-Förderer, der die Ver- und Entsorgung der Station mit Werkstücken übernimmt. Als Koppelglieder zwischen dem Regalförderzeug und den BS-Förderern werden fünf Drehtische eingesetzt. Auf jedem Drehtisch sind zwei Werkstückplätze vorgesehen, die als Bereitstellungsplätze für Bearbeitungsstationen bzw. des RFZ dienen. Für die Durchführung der Drehbewegung der Drehtische wurde eine unterlagerte Steuerung mit einer Schnittstelle zum Prozeßrechner aufgebaut. Zwischen den Drehtischen sind parallel zur RFZ-Schiene Rollenbahnstücke angebracht (Transferbahnen), die den Transport eines Werkstückes zwischen zwei benachbarten Bearbeitungsstationen erlauben und damit zur Entlastung des Regalförderzeugs beitragen. Die Bearbeitung an den Bearbeitungsstationen wurde im Modell nur andeutungsweise simuliert. Dazu besitzen die Bearbeitungsstationen eine autonome Steuerung. Die fertigbearbeiteten Werkstücke werden vom Regalförderzeug an der Auslagerungsstation (AS) abgegeben. Von dort gelangen die Werkstücke durch eine Absenkvorrichtung zu den Rollenbahnen des Rüstbereiches.

Durch diese Materialflußkonzeption ist ein Kreislauf der Werkstücke gewährleistet. Die Anlage kann hierdurch ohne einen externen Eingriff ununterbrochen arbeiten.

1.2 Die Software - Umfang und Grobstruktur

Das Steuerungsprogramm wurde in Moduln aufgeteilt, die den fördertechnischen Bereichen entsprechen. Zusätzlich findet in einem Modul die Disposition der Transportaufträge statt und für die Anfangsbedingungen beim Start des An-

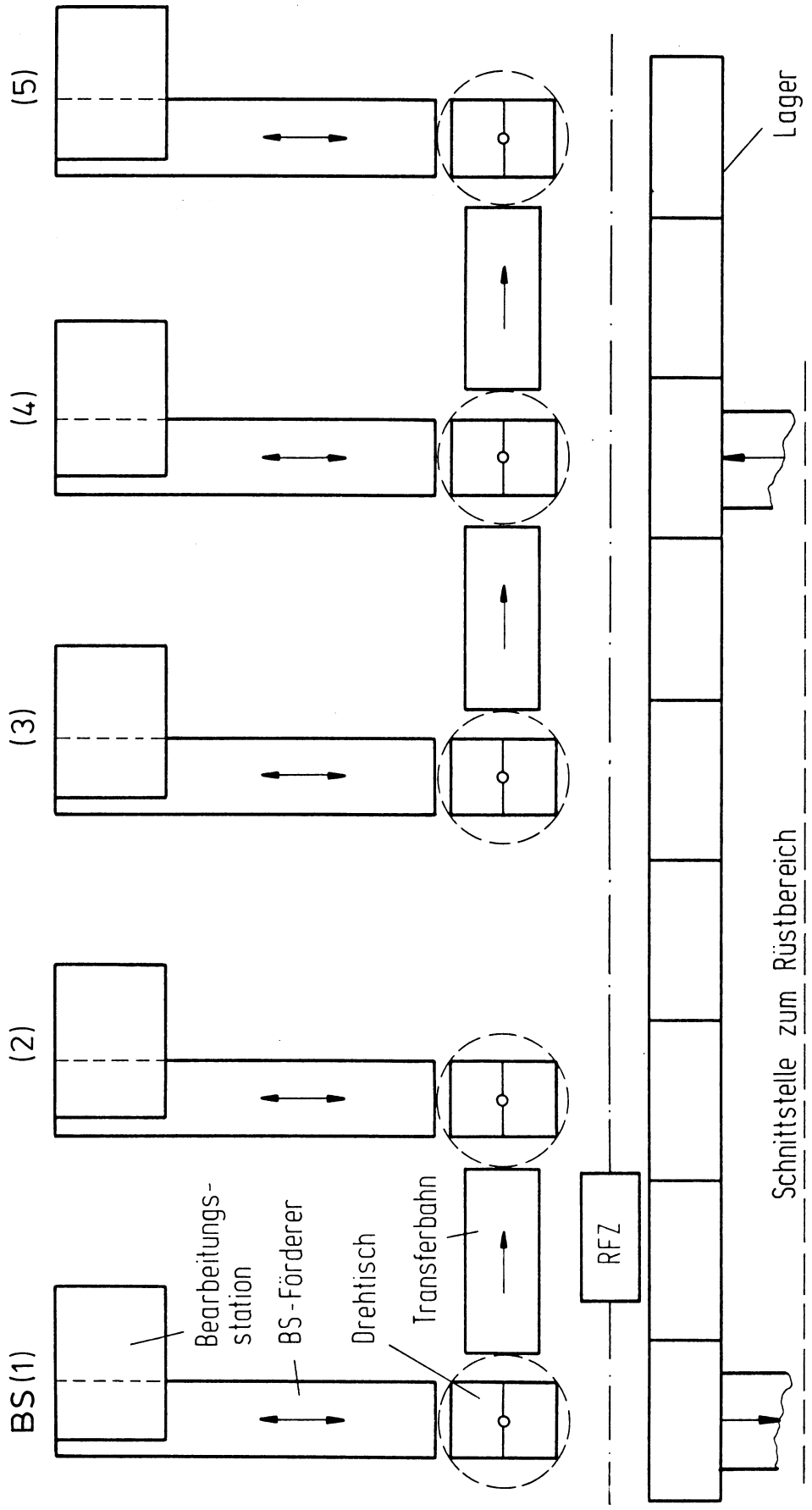


Bild 2: Fertigungsbereich

wendersystems wurde ein Initialisierungsmodul geschaffen. Das Anwenderprogramm besteht somit aus folgenden Moduln:

1) Rüstbereich
2) Regalförderzeug (RFZ)
3) Fertigung
4) Disposition
5) Initialisierung

Die Moduln beinhalten insgesamt 40 Tasks (20 k-Bytes Zielcode).
An den Prozeßrechner sind 80 Interrupts und 80 Digitalausgaben angeschlossen. Es werden 60 Motoren durch den Prozeßrechner gesteuert.

Die drei ersten Moduln steuern die entsprechenden fördertechnischen Einrichtungen. Der Dispositionsmodul verwaltet das Lager und die Transportaufträge für das RFZ. Er beinhaltet ebenfalls die Beauftragung des RFZ und kontrolliert die Ausführung der Transportaufträge.

Der Initialisierungsmodeul schafft einen genormten Anfangszustand (globale Anfangsbedingungen) beim Starten des Anwendersystems. Er aktiviert die Initialisierungstasks für lokale Anfangsbedingungen in den übrigen Moduln.

2. Charakteristische Merkmale von Förderprozessen in der Fertigung

Der Einsatz eines Prozeßrechners zur Steuerung des Materialflusses ist erst bei einer gewissen Komplexität des Fördersystems gerechtfertigt. Die hierfür in Frage kommenden Fördersysteme in der Fertigung sind gekennzeichnet durch:

1) eine große Zahl von Förderstrecken, in denen die Existenz von Fördergütern (Werkstücken) zwecks Materialflußverfolgung kontrolliert werden muß,
2) eine Vielzahl von Weichen, die in ihrer Stellung kontrolliert und gegebenenfalls geschaltet werden müssen und
3) eine große Zahl von Antrieben, die stationär an den Förderstrecken bzw. Weichen angebracht sind oder in Fahrzeugen mitgeführt werden.

Außerdem ist allen Förderprozessen gemeinsam, daß sich an Zusammenführungen bzw. nacheinander folgenden Strecken mit unterschiedlichem Durchsatz Warteschlangen von Stückgütern bilden können. Warteschlangen können aber auch in Form von Transportanforderungen entstehen, die die Benutzer an ein Fördermittel (wie z.B. Regalförderzeug) stellen. Die Warteschlangen an den Förderstrecken sind konstruktionsbedingt nach dem first in - first out Prinzip aufgebaut. Bei Warteschlangen von Transportaufträgen können dagegen auch

andere Ordnungsprinzipien Anwendung finden.

3. Konsequenzen für das Steuerungssystem

Aus den Bedingungen der Förderprozesse ergeben sich einige Folgerungen bezüglich der Gestaltung der Software des Steuerungssystems. Vor allem ist ein Abbild des Materialflusses im Prozeßrechner zu führen, das bei Änderungen der Transportsituation ständig aktualisiert werden muß.

Die hohe Anzahl der Förderstrecken und die hierdurch bedingte große Zahl von Endschaltern bewirkt, daß zur Meldung von Fördergutpositionen entsprechend viele Interrupts vom Rechner verarbeitet werden müssen. Dabei sind die nach diesen Meldungen jeweils zu erfüllenden einzelnen Aktualisierungsaufgaben untereinander sehr ähnlich. Durch den Einsatz vom Prozessperipherieeinheiten, deren Eingänge zyklisch abgefragt werden, kann die Zahl der Interrupts gesenkt werden. Nachteilig bei dieser Lösung ist, daß außerhalb der Spitzenbelastung des Materialflußsystems der Prozeßrechner häufig die Auswertung der eingelesenen Daten durchführt, die unverändert bleiben und keine Änderung des Materialflußabbildes bewirken.

Somit ist es zweckmäßig, zur Steuerung des flexiblen Fertigungssystems als Prozeßinterface die Prozeßunterbrechungseinheiten (PUE) einzusetzen.

Die Disposition der Transportanforderungen und die Wahl des Förderweges erfordern Zugriffe zu dem Materialflußabbild. Diese müssen synchronisiert werden, um zu vermeiden, daß ein Fördergut aufgrund nicht synchronisierter Abfragen z.B. zu zwei verschiedenen Zielen gleichzeitig verplant wird. Die Zeitpunkte, wann Transportanforderungen von den Fördermittelbenutzern gestellt werden, sind nicht voraussagbar. Ebenso kann der Fall auftreten, daß einige Anforderungen gleichzeitig gestellt werden. Deswegen mußten diese Anforderungen auch als Interrupts behandelt werden, die durch Software getriggert wurden.

4. Problemstellungen für die Software und ihre Lösung in PEARL

Zur Verfolgung der Fördergüter im Materialfluß müssen die Warteschlangen für die Datensätze der Fördergüter im Kernspeicher nachgebildet werden. Dazu eignet sich der Aufbau von Listen. Für den Listenaufbau, vor allem für Elemente, die im Kernspeicher verstreut sind, können Referenzen (Pointer) benutzt werden. Die Referenzen erlauben eine leichte Verkettung der Listenelemente und einen schnellen Zugriff, da die Operationen nur mit Adressen (Pointern) ausgeführt werden müssen.

Zur Bearbeitung dieser Listen können Prozeduren formuliert werden, die die Zugriffe für das Einschreiben (hinter dem letzten Listenelement) und Ausschreiben (Zugriff zum ersten Listenelement) realisieren. Die Verwaltung von Transportaufträgen kann mehr Aufwand benötigen, wenn die Aufträge wie im vorliegenden Fall verschiedene Prioritäten haben, die auch andere Warteschlangenzugriffe als nur zum letzten und ersten Element erfordern. In diesem Falle läßt sich die Auftragsliste in mehrere Teillisten spalten, die jeweils die Aufträge gleicher Priorität verwalten. In den Teillisten sind dann nur die Zugriffe zum ersten und letzten Element notwendig. Damit ist das Problem auf einfache Warteschlangenzugriffe zurückgeführt.

Die FIRST {Element} und LAST {Element} Zugriffe für Warteschlangen sind im Förder- und Lagerwesen sehr wesentlich, da die Menge aller Warteschlangen von Fördergut-Datensätzen im Rechner das Materialflußabbild darstellt.

Die Änderung der Daten im Materialflußabbild durch verschiedene Tasks erfordert die Anwendung der Synchronisationsvariablen (Semafore).

Die Semas erlauben den Schutz vor unberechtigten Zugriffen zu den Daten, der einen Ausschließlichkeitscharakter hat (bei Sema Werten 0,1) oder einen begrenzten parallelen Zugriff zu Daten erlaubt (Sema Werte $>$ 1). Sollte der Schutz nur einen ausschließlichen Zugriff zulassen, muß die Sema-Benutzung sehr sorgfältig vorgenommen werden. Eine RELEASE-Anwendung mehr in irgend einer Programmverzweigung, die sehr selten bei gewissen Konfigurationen des technologischen Prozesses durchlaufen wird, setzt nach diesem einmaligen Durchlauf die Synchronisation völlig außer Kraft. Dieser Fehler ist besonders gefährlich, da die geänderte Situation nach außen hin nicht sichtbar wird. Der Fehler im Programm kann hierdurch sehr lange unentdeckt bleiben. Falls allerdings eine REQUEST-Anweisung zweimal nacheinander durchlaufen wurde, unterbricht der Rechner die weitere Arbeit, falls ausschließliche Zugriffe festgelegt wurden. Damit aber wird der technologische Prozeß vom Rechner nicht mehr kontrolliert, wogegen bei der Planung der Anlage entsprechende Absicherungen vorgesehen werden müssen. Eine Vielzahl von Interrupts, die der Prozeßrechner verarbeiten muß, kann die Anzahl der Tasks im Anwendersystem in die Höhe treiben. Um die Taskanzahl in Grenzen zu halten bzw. zu reduzieren, kann die Zusammenfassung einiger Taskcodes während der Programmentwicklung zu einer größeren Task angestrebt werden. Dazu müssen diese Interrupts in einer Einplanungsliste für die neue Task zusammengefaßt werden.

Dies bedingt aber die Erfüllung folgender Voraussetzungen:
Sollten im ungünstigsten Fall zu einem Zeitpunkt alle Interrupts dieser Liste vom Prozeß an den Rechner gemeldet werden, darf die zulässige Abarbeitungszeit für jeden dieser Interrupts nicht überschritten werden. Andernfalls ent-

steht die Situation, daß der Prozeßrechner mit dem Prozeß nicht schritthalten kann. Eine weitere Voraussetzung für die Straffung des Anwenderprogramms in Bezug auf die Taskzahl schafft die Prozedur ORIGIN. Durch den Einsatz dieser Standardprozedur läßt sich der beauftragende Interrupt eines Tasklaufs numerisch identifizieren. Das kann zum Verzweigen in die entsprechenden Codeteile der Task benutzt werden, die die Antwort des Anwenderprogramms auf diesen Interrupt darstellen. Viele Teilaufgaben (z.B. bei der Aktualisierung des Abbilds der Förderstrecken) sind sich sehr ähnlich. Deswegen kann durch Zusammenfassung mehrerer Interrupts in der Einplanung einer Task nicht nur zur Einsparung der Taskanzahl, sondern auch des Kernspeicherbedarfs für den Taskcode selbst beigetragen werden.

Die Interrupts, die die Transportanforderungen anmelden oder die Aktualisierung des Materialflußabbildes veranlassen sollen, können auch während des Ablaufs der zu beauftragenden Task eintreffen. Diese Folgeaktivierung wird solange gepuffert, bis die Taskaktivität beendet ist. Dann werden die Folgeaktivierungen sequentiell abgearbeitet. In anderen Sprachen wie z.B. Prozeß FORTRAN können Folgeaktivitäten nicht verarbeitet werden. Sie führen entweder zur nicht Beachtung der Folgeaktivierung oder zum Abbruch der Programmausführung.

Bei der Disposition von Förderaufträgen entstehen oft Situationen, die zu Folgeaktivierungen führen. Wenn eine Task die Transportanforderungen von verschiedenen Stellen annimmt und verarbeitet, sind die Zeitpunkte, in denen die Anforderungen gestellt werden, nicht voraussagbar. Es können Fälle auftreten, daß mehrere Transportanforderungen gleichzeitig gestellt werden, das heißt, mehrere Aktivitäten von der verarbeitenden Task gefordert werden. In dieser Konstellation entstehen durch Pufferung der Interrupts und sequentieller Abarbeitung der Anforderungen keine Probleme. Nichtvorhandensein der Pufferung von Taskbeauftragungen bringt die Notwendigkeit des Aufbaus der Interruptspuffern im Anwenderprogramm mit sich, was die Programme verkompliziert und möglicherweise neue Fehlerquellen entstehen läßt.

5. Schlußbemerkungen

In der Zukunft ist mit einer Zunahme der Automationsaufgaben zu rechnen, um die ständig wachsende Kompliziertheit der Fertigungsvorgänge und des damit verbundenen Materialflusses sicher beherrschen zu können. Die Sprachen für die Prozeßsteuerung bringen eine Erleichterung in der Entwicklungs- und Testphase der Programme. Ebenfalls verbessert sich ihre Selbstdokumentation. Diese Gründe werden zum verstärkten Einsatz der höheren Prozeßprogrammiersprachen, insbesondere PEARL, bei der Bewältigung der Automationsaufgaben führen.

Die Automatisierung eines industriellen Entwicklungslabors mit PEARL

R. Barth, G. Bauknecht GmbH, Stuttgart

1. Einleitung

In den Zentrallabors der G. Bauknecht GmbH in Stuttgart werden die Entwicklungsprüfungen und Versuche an Hausgeräten vom ersten Prototyp bis zum serienreifen Gerät durchgeführt.

Neben diesen Versuchen finden aber auch Approbationsprüfungen sowie Nachprüfungen an reklamierten Kundengeräten statt.
Die Versuche, die in diesen Entwicklungslabors ablaufen, haben also weniger den Charakter einer starren Prüfung mit festem Ablauf, sondern werden mehr empirisch den experimentellen Anforderungen angepaßt.
Dies ist im wesentlichen auch der Grund, weshalb die Prozeßdatenverarbeitung als Hilfsmittel zur Rationalisierung der Versuche bisher noch nicht angewandt werden konnte, und fast ausschließlich mit der herkömmlichen Meßtechnik, wie z. B. Mehrkanalschreibern, anzeigende Analoginstrumente und Zähler gearbeitet wurde.

Erst als große Kapazitätsprobleme auftraten, die ausgelöst wurden durch kaum mehr zu bewältigende Datenmengen, das immer weiter sinkende Qualifikationsniveau des Laborpersonals und die räumliche Begrenzung der Prüfkabinen, war man gezwungen, den Laborbetrieb durch Automation zu rationalisieren. Dies setzte jedoch voraus, daß man von dem bislang praktizierten absolut freizügigen Experimentieren abkommen - und die Versuche in gleichartigen Prüfabläufen integrieren mußte.

Trotz des Versuchs, die Prüfungen in wenigen Standardabläufen zu vereinigen, ergab die nach wie vor bestehende Forderung nach größtmöglicher Flexibilität bei der Gestaltung der Prüfungen eine Vielzahl von unterschiedlichen, aus Standardabläufen zusammenstellbaren Versuchsvarianten.
Diese Versuchsumstrukturierung wurde bereits frühzeitig durchgeführt, so daß das Personal die neuen Versuchsabläufe mit der konventionellen Meßtechnik praktizieren konnte.

2. Die Anforderungen an das automatische Laborsystem

Betrachtet man den bisherigen Ablauf und die Organisation der Laborversuche, so stellt man fest, daß durch den völlig freien Einsatz von schreibenden und anzeigenden Meßinstrumenten für Temperaturen, Drücke, Ströme und Spannungen sowie von Zählern für die elektrische Arbeit, der Meßtechniker die Versuche absolut flexibel gestalten konnte.
Dasselbe gilt für die Anschlußmöglichkeiten der Meßwertgeber in den Prüfkabinen. Aus diesem Grund war natürlich die Forderung nach größtmöglicher Flexibilität der oberste Grundsatz für die Auslegung der Hard- und Software für das neue automatische Laborsystem.

Im einzelnen wurde folgendes gefordert:

- Jeder Prüfling muß mit einer variablen Anzahl (bis zu 40) von Meßwertgebern ausgerüstet werden können.

- Jeder Prüfling muß an jeden freien Prüfplatz in jeder beliebigen Prüfkabine aufgestellt werden können.

- Die Meßwertgeber müssen an jeder freien Buchse des entsprechenden Gebertyps in den Prüfkabinen angeschlossen werden können.

- Jeder Prüfling muß mit einer variablen Spannung versorgt werden können.

- Die Versuche müssen zu jeder Zeit unabhängig voneinander gestartet - und abgebrochen werden können.

- Eine beliebige Anzahl von Prüflingen muß mit gleichartigen Versuchsabläufen zur gleichen Zeit simultan geprüft werden können.

- Jede Kombination und Anzahl von Prüfabläufen muß für jeden Prüfling zulässig sein.

Es leuchtet ein, daß dieser hohe Flexibilitätsgrad an die Programme zur Organisation der Versuche sehr hohe Anforderungen stellt.
So sind sehr viele Abprüfungen auf Zulässigkeit, Plausibilität und Verträglichkeit sowie eine große Anzahl von Zuordnungen innerhalb der Prüflings-, der Prüfplatz- und der Meßbuchsenorganisation durchzuführen.

Desgleichen wird die gesamte Prüfplatz- und Prüflingsverwaltung recht aufwendig. Die Forderung nach Flexibilität der Prüfabläufe setzt ebenso eine entsprechend komplizierte und sichere Programmorganisation voraus, wobei hier noch erschwerend das Echtzeitverhalten der entsprechenden Programme hinzukommt.

Eine weitere Forderung der Anwender betraf die Verarbeitung von Verbalinformationen. Angefangen bei den Prüflingen, über die Meßstellenarten bis hin zur Darstellung der Prüfergebnisse und der Versuchsergebnisarchivierung, wurde bereits in der konventionellen Laborорganisation zur Klassifizierung Namen, Texte, Stichworte und alphanumerische Bezeichnungen verwendet.
Von diesen selbsterklärenden und bereits seit langem eingeführten und bewährten Bezeichnungen wollte und konnte man bei einem automatischen Laborsystem nicht abgehen. Das neue Laborsystem mußte deshalb eine komfortable Textverarbeitung zulassen.

Es war klar, daß diese Anforderungen insbesondere bezüglich der Echtzeitverarbeitung als auch der Textverarbeitung in dem gegebenen Umfang sehr hohe Anforderungen an die einzusetzende Programmiersprache als auch an das Betriebssystem stellen.
Eine Programmierung auf Assemblerebene schied deshalb von vornherein aus. Der Kreis der gängigen höheren Programmiersprachen war auch sehr schnell eingeengt, da im wesentlichen nur PEARL bzw. PAS 2 (BBC-PEARL) die beiden Voraussetzungen auf befriedigende Weise erfüllen.

Auch bezüglich spezieller Hardwareeigenschaften waren Anforderungen der Anwender zu erfüllen. So sollte die bisher bestehende Unabhängigkeit der einzelnen Labors untereinander auch bei einem rechnergesteuerten Laborsystem gegeben sein.
Desweiteren sollte in allen Labors eine direkte Dialogmöglichkeit mit dem System bestehen.
Die Versuchsergebnisse sollten bei allen Anwendungen in Form von Tabellen mit Zahlenwerten und erläuterndem Text, als auch in graphischer Form als Kurvenschaaren bzw. Diagramme ausgegeben werden können.
Um Auswertungen von größeren, auch länger zurückliegenden Versuchsreihen zu ermöglichen, sollten die Versuchsdaten, einschließlich aller relevanten Beschreibungen, im Laborsystem maschinell lesbar abgespeichert werden.
Ein sehr wesentliches Kriterium betraf die Meßgenauigkeit von Kleinstspannungen. Insbesondere aufgrund der Anforderungen in den genormten Prüfvorschriften für Hausgeräte, die besonders bei Temperaturmeßwerten nur einen maximalen Fehler von $\pm$ 0,3°C zulassen, wurde die Hardwareauswahl stark beeinflußt.

Neben all diesen Anforderungen speziell aus den Labors waren natürlich noch weitere Kriterien, die mehr die Systembetreuung betrafen, zu erfüllen.

3. Die realisierte Prozeßrechnerstruktur

Aufgrund der Anforderungen, die an die Hardware gestellt wurden, war klar, daß in jedem Labor ein verhältnismäßig "intelligentes" Rechnersystem, das weitestgehend autark und unabhängig von den anderen Systemen ist, zu installieren war. Andererseits erforderte die Ergebnisdarstellung und auch die Versuchsarchivierung eine verhältnismäßig teure Peripherie, die wiederum im jeweiligen Labor für sich alleine nicht genügend ausgelastet worden wäre.
Es lag deshalb nahe, die intelligenten dezentralen Rechnersysteme mit einem gemeinsamen, übergeordneten Rechnersystem zu verbinden. Dort konnte dann die für alle Labors gemeinsam nutzbare, teurere Peripherie wie z. B. Lochkartenleser, Zeilendrucker, Magnetband, Graphikbildschirm und Plotter angesiedelt werden. Hiermit stand auch für die Softwareentwicklung ein ausreichend komfortables Rechnersystem zur Verfügung.

Bei diesem nun entstandenen hierarchischen Rechnerkonzept (Abb. 1) war für die Auslegung der Satelliten-rechner und der Übertragungsleitungen nur noch die Frage entscheidend:
"An welcher Stelle in der Verarbeitungskette der Software wird sinnvollerweise die Schnittstelle zwischen Satelliten- und Zentralrechner gelegt?"

Die Antwort hierauf ergab sich bereits aus den beiden Randbedingungen, und zwar einerseits, daß das Gesamtsystem Labor für Labor stufenweise ausgebaut werden soll und, daß andererseits unter dem gewählten PAS 2 - Echtzeitbetriebssystem ein Foreground - Backgroundbetrieb am Zentralrechner nicht möglich ist.

Das hatte zur Konsequenz, daß die einzelnen Laborrechner die jeweiligen Aufgaben, mit Ausnahme der komfortablen Versuchsauswertung und der Versuchsarchivierung, komplett eigenständig übernehmen mußten. Hierfür war an Rechnerhardware neben einer leistungsfähigen Zentraleinheit, einer bzw. mehrerer Bedienkonsolen auch ein Plattenspeicher für die Abspeicherung der nicht residenten Anwenderprogramme, der Parameter- und Textdateien und der Meßdaten notwendig.

Eine weitere Zergliederung der Rechnerhierarchie auf der unteren Laborebene aus Gründen der Ausfallsicherheit hat sich wegen den hohen Kosten als nicht sinnvoll erwiesen.
Dies insbesondere deshalb, weil auf der einen Seite die Anforderungen an die Ausfallsicherheit in dieser Anwendung nicht so hoch sind wie beispielsweise in einem Prüffeld der Fertigung, und andererseits, weil die Anwendersoftware in den Systemen innerhalb eines jeden Labors identisch aufgebaut gewesen wären und somit die gleichen Systeme wie im jetzigen Konzept lediglich mehrfach vorhanden gewesen wären.

4. Die Konzeption der Anwendersoftware

Die Konzeption der Software soll im nachfolgenden am Beispiel des Kühl- und Gefriergerätelabors näher erläutert werden.
Gerade in diesem Labor stand die Flexibilität bezüglich der variablen Aufstellung der Prüflinge und des freizügigen Anschlusses der Meßwertgeber im Vordergrund.
Außerdem waren hier auch für alle 512 Analogeingänge der Prozeßperipherie die höchsten Genauigkeitsanforderungen gestellt worden. Den schematischen Aufbau der Prüfräume und der Prüflingsanschlußmöglichkeiten zeigt Abb. 2.

In jedem der sechs Prüfräume sind an den Wänden verteilt jeweils 80 Meßbuchsen für Thermoelemente angebracht. Hinzu kommen jeweils 4 Meßbuchsen für Druckgeber. Die Steckdosen (6 Stück pro Prüfkabine) jedes Prüfplatzes können mit variablen Spannungen versorgt werden, die am Laborrechner einstellbar sind.
Jeder Abgang zu den einzelnen Prüflingen wird mittels Strom- und Leistungswandlern überwacht.

Der vorliegende Aufbau erlaubt damit den Meßtechnikern, jeden Prüfling mit bis zu 40 Thermoelementen bzw. mit bis zu zwei Druckgebern zu bestücken. Die Aufstellung und der Anschluß der Prüflinge an das Laborsystem kann ausschließlich unter dem Gesichtspunkt der Belegung von freien Plätzen erfolgen. Damit ist gewährleistet, daß die Prüfräume und auch das Laborsystem optimal ausgelastet werden kann. Diese Flexibilität drückt sich natürlich auch in der Prüfplatzbeschreibung innerhalb der Anwendersoftware aus. Hierauf wird weiter unten näher eingegangen.

Dieser wird als neue Zyklusvorgabe der Zyklussteuerung mitgeteilt, die dann nach Ablauf der Zyklusvorgabe die entsprechende Funktionstask für die jeweiligen Meßstellen erneut aktiviert.
Mit diesem Konzept ist es möglich, daß ein und dieselbe Funktionstask mehrere Prüflinge, die sich alle in verschiedenen Versuchsstadien befinden können, quasi simultan überwacht. Die Strategie der variablen, aus den letzten Meßwerten abgeleiteten Abfragezeit, wurde deshalb gewählt, weil die Meßwertverläufe an den Prüflingen sehr ungleichmäßig sind. So hätte man bei einem festen Abfragezyklus das Zeitraster an der schnellstmöglichen Meßwertänderung orientieren müssen, wobei dann bei langsamen Meßwertänderungen der ohnehin kleine Speicher mit unwichtigen Meßwerten gefüllt worden wäre.
Entsprechend wie bei der Zyklussteuerung melden sich die meisten Funktionstasks beim Erreichen des von dieser Funktionstask zu überwachenden Versuchszustandes bei der Ablaufsteuerung zurück. Diese wiederum prüft, ob bereits alle in dieser Versuchsphase, z. B. von anderen Funktionstasks erwarteten Rückmeldungen eingetroffen sind, und schaltet im positiven Fall auf die nächste Versuchsphase um. In dieser neuen Phase kann sowohl der Start neuer Funktionen, als auch der Weiterlauf von bereits in der Vorphase aktiven Funktionen notwendig sein.
Entsprechend den Erfordernissen des Versuchsablaufs steuert dann die Versuchsablaufsteuerung die jeweiligen Funktionstasks an.

Neben der Aufgabe, die Meßwerte des Prüflings nach bestimmten Strategien zu erfassen und das Erreichen eines bestimmten erwarteten Ereignisses an die Ablaufsteuerung zurückzumelden, haben die Funktionstasks die Aufgabe, die Meßwerte an die entsprechenden Dateien weiterzuleiten, um sie dort unter dem jeweiligen Versuchs-Auftrag abzulegen.

In Abb. 3 ist die Softwarekonzeption, wie sie für die Satellitensysteme realisiert wurde, dargestellt. Diese gliedert sich in das Online-System und das Dialog-System. Beide werden verbunden über das gemeinsame Dateiensystem.
Das Dialogsystem, das aus zwei residenten Teilen und 16 Overlaysegmenten aufgebaut ist, dient einerseits in steuernder Funktion zur Versuchsinitialisierung, zum Versuchsstart und zum Versuchsabbruch sowie andererseits für die Versuchsinformation in Form von Zwischenprotokollen, Zustandsbeschreibungen und Ergebnisausgaben. Mit Hilfe dieses Dialog-Systems werden also die gesamten Zuordnungen innerhalb der Prüfplatzbeschreibung und die Versuchsbeschreibung vom Meßtechniker aufgebaut. Für die Informationsausgabe über den Versuch selbst und auch über den gesamten Prüfkabinenstatus, wie z. B. Prüfplatz- oder Meßbuchsenbelegung, dienen die Informationsdialoge.

Das Online-System, dessen Hauptaufgabe die Meßwerterfassung und Versuchssteuerung bzw. -überwachung ist, gliedert sich in einen allgemeinen Meßwerterfassungsteil und in die spezifischen Funktionstasks, die von der Versuchsablaufsteuerung bzw. Zyklussteuerung über einen Auftragspuffer aktiviert werden. Die einzelnen Funktionstasks, die abhängig vom jeweiligen Versuch bzw. vom jeweiligen Versuchsstatus von der Ablaufsteuerung aktiviert werden, rufen die entsprechenden Meßwerte nicht in einem festen Zeitraster zyklisch ab, sondern ermitteln selbst nach jedem Einlesen eines Meßwertes aufgrund verschiedener Kriterien den nächsten Einlesezeitpunkt.

Dieses Dateienkonzept, das schematisch in Abb. 4 dargestellt ist, enthält neben den erforderlichen statischen Basisdaten, die Beschreibungen der Prüfplätze und der Versuche.
Gerade wegen der flexiblen Gestaltung der Versuche ist die Verkettung der Versuchsbeschreibung und der Prüfplatzbeschreibung recht aufwendig. Außer der Komplexität der Verkettung ist auch das Dateienkonzept mengenmäßig verhältnismäßig großzügig angelegt. So sind gleichzeitig bis zu 999 Versuchsaufträge, die jeweils bis zu 9 Prüflinge umfassen können, zulässig. Mit jedem dieser Prüflinge können bis zu 512 Einzelversuche durchgeführt werden.
Innerhalb der Prüfplatzbeschreibung erfolgt die logische Verknüpfung zwischen dem Prüfling, dem Prüfplatz, und den belegten Meßbuchsen. Nur die Meßbuchsen selbst sind fest mit der Hardwarebeschreibung verbunden.
Die Versuchsbeschreibung beinhaltet neben den zur Organisation bzw. Klassifizierung eines Versuchs notwendigen Beschreibungen, wie z. B. die Versuchsauftragsnummer, den Geräte-Typ, die Prüflingsnummer und die Versuchsnummer, noch weitere Informationen. Dieses sind im wesentlichen Verbalinformationen, die über den Gerätebauzustand, den verantwortlichen Meßtechniker usw. Auskunft geben.
Als Suchkriterien für das Wiederauffinden von archivierten Versuchen, wurde ein Stichwortkatalog eingeführt. Darin sind von jedem Versuch einige, den Versuch kennzeichnende Kriterien abgelegt.

5. Erfahrungen mit PEARL

Stellt man sich nun nach der erfolgreichen Inbetriebnahme der ersten beiden Ausbaustufen die Frage, ob die Entscheidung, das Projekt mit PEARL (bzw. PAS 2) zu realisieren, richtig war, so muß man diese, objektiv betrachtet, eindeutig bejahen.
Es war natürlich von Anfang an klar, daß der Einsatz einer so jungen Sprache mit großen Risiken verbunden ist. Auch wir mußten deshalb einiges an Lehrgeld bezahlen.
So fielen unsere Speicherabschätzungen, die auf Erfahrungen mit Assemblersprachen und FORTRAN beruhten, fast durchweg zu klein aus. Dies hatte zur Folge, daß die zunächst vorgesehenen kleineren Satellitenrechner gegen größere ausgetauscht werden mußten. Auch die Hardwareauswahl wurde von der gewählten Sprache sehr beeinflußt, da die Anzahl von bereits implementierten Geräten nicht so groß ist, wie beispielsweise bei Verwendung von Standardherstellersoftware. Aus diesem Grunde mußte beispielsweise für die Graphikausgabe auf einem Plotter diesem ein in BASIC programmierbarer Tischrechner, der die Graphik-Software enthält, vorgeschaltet werden.
Da beide Nachteile - großer Speicherbedarf und eingeschränkte Hardwareauswahl - sich im Laufe der Zeit, z. B. durch bessere Codeoptimierung bzw. die Implementierung weiterer Geräte, beseitigen lassen, ist diesen Einschränkungen nur vorübergehende Bedeutung beizumessen.
Trotzdem sollte hieran intensiv gearbeitet werden.

Was die Erlernbarkeit der Sprache anbetrifft, so stellten wir fest, daß bezüglich der reinen Batch-Programmierung zu anderen, höheren Programmiersprachen kaum ein Unterschied besteht.
Die Schwierigkeiten treten erst dann auf, wenn das volle Repertoire der Echtzeitprogrammierung im Multitaskbetrieb beim Einsatz der Overlay-Technik Verwendung findet.
Da hierbei in großem Umfang die Möglichkeiten des PEARL-Betriebssystems ausgeschöpft werden, ist es für den Programmierer notwendig, diese internen Zusammenhänge zu verstehen. Um diese Schwierigkeiten kommt der Programmierer aber auch nicht bei der Assemblersprache herum. Insofern ist PEARL, was die Erlernbarkeit anbetrifft, nicht anders zu sehen als beispielsweise FORTRAN oder PL 1 mit Echtzeit- und Multitask-Erweiterungen.

Bezüglich des Sprachumfangs bzw. des Komforts stellten wir außer dem Fehlen von Graphikaufrufen, keine Mängel fest.
Ein in dieser Anwendung gravierender Nachteil des Softwaresystems ist jedoch die fehlende Eigenschaft, während des Echtzeitbetriebs im Hintergrund zu compilieren.
Dies ist aber kein Mangel der Programmiersprache selbst, sondern betrifft ausschließlich die Eigenschaften des Betriebssystems und des Compilers. Dieser Mangel des BBC-PEARL-Programmsystems sollte in jedem Fall behoben werden. Trotz dieser kleineren Unzulänglichkeiten, ist das BBC-PEARL-System ein Softwaresystem, das bereits einen hohen Grad an Ausgereiftheit erreicht hat. Man kann deshalb ohne Einschränkung feststellen, daß die PEARL-Entwicklung sinnvoll und auch erfolgreich war.

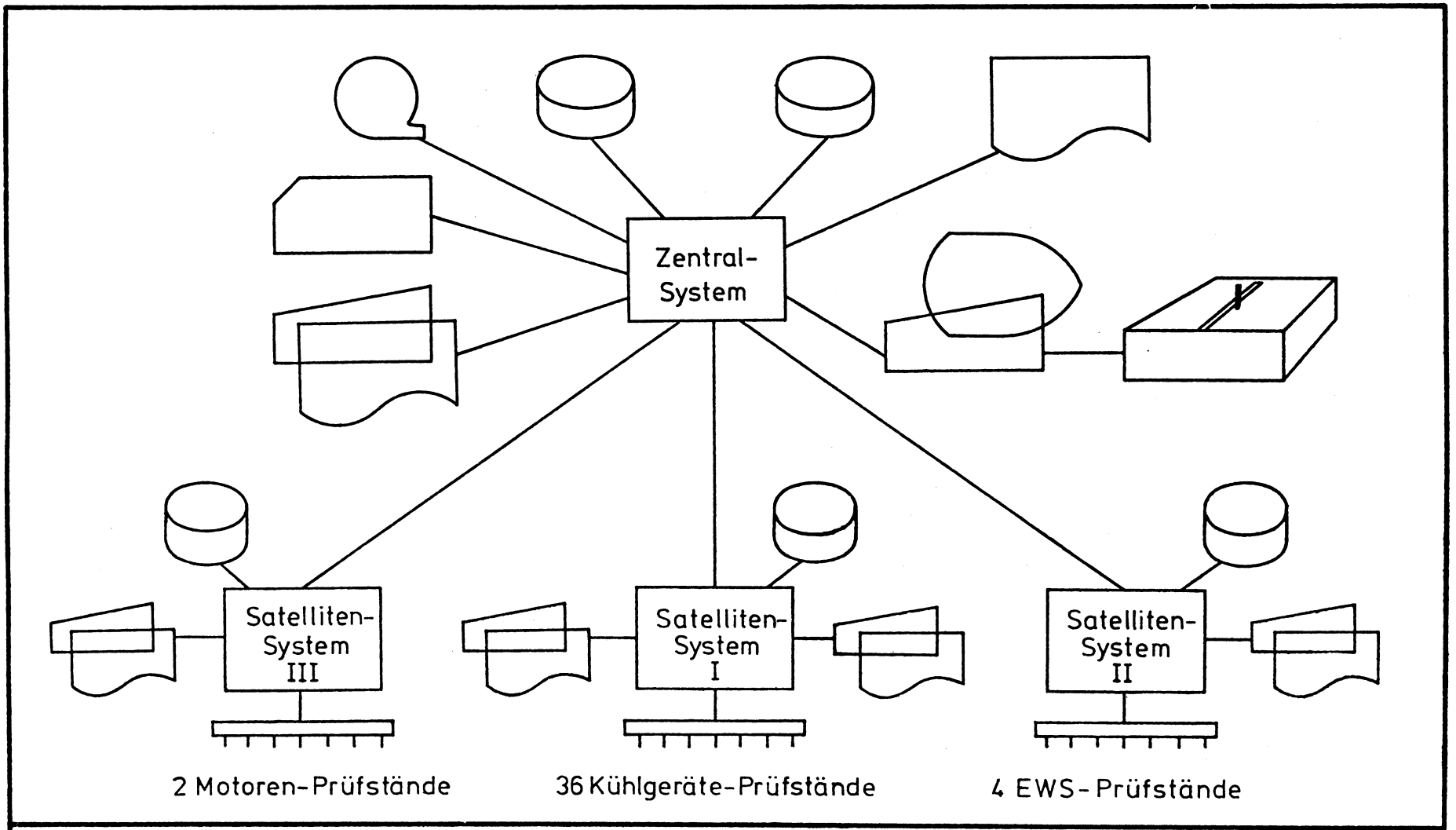


Abb.1 Hardwarekonfiguration des Laborsystems

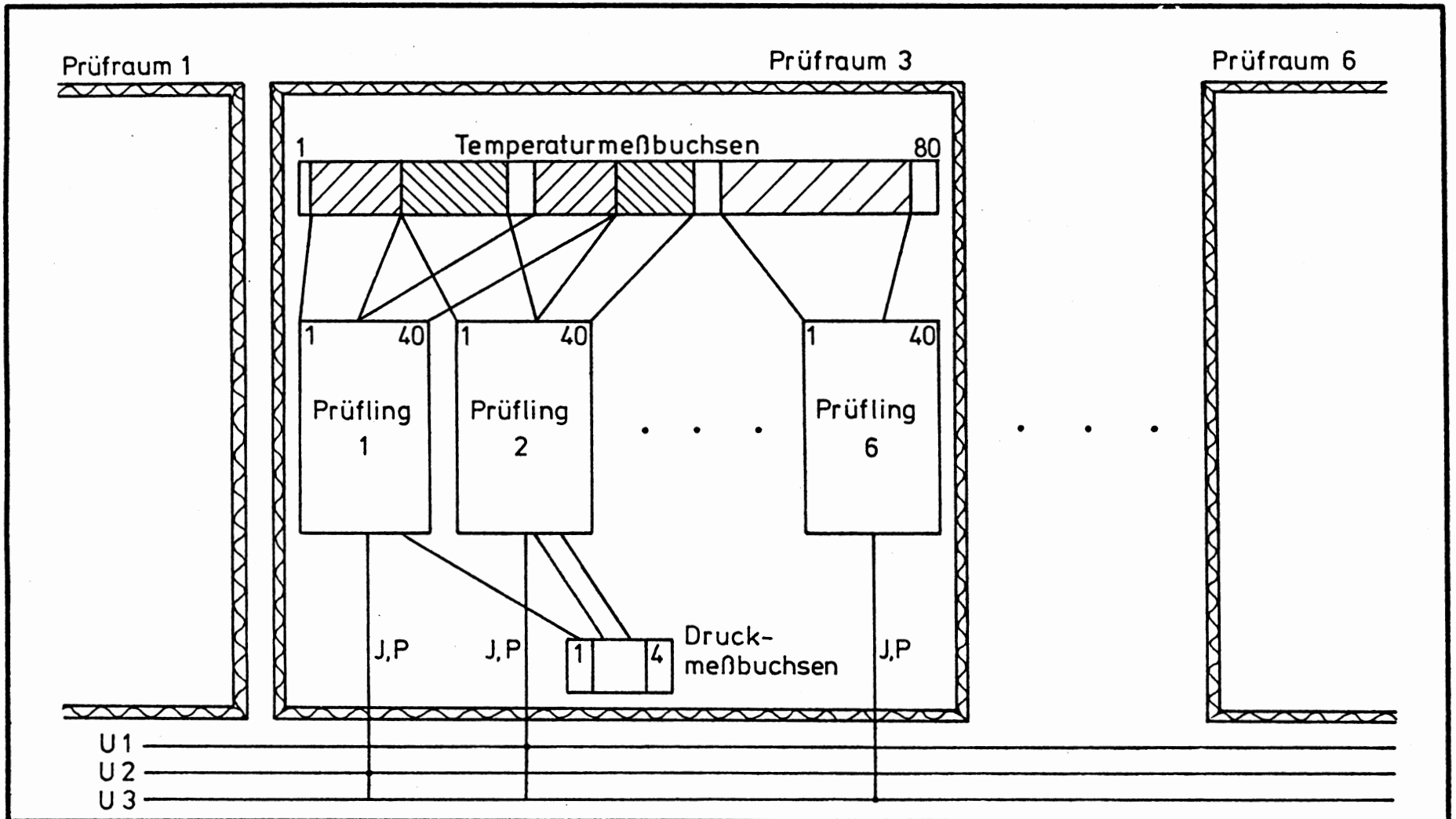


Abb.2 Schemadarstellung der Kühlgeräte-Prüfräume

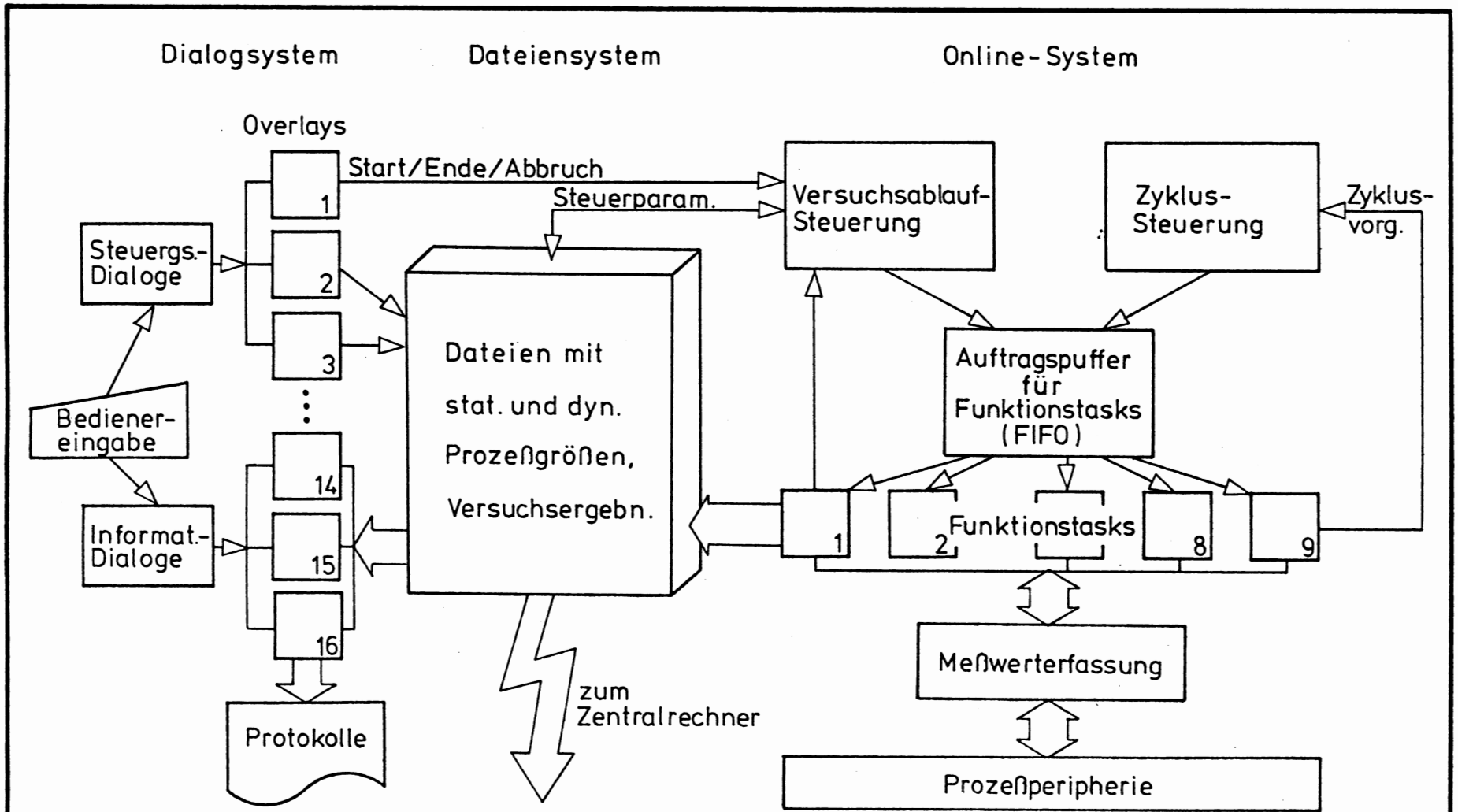


Abb. 3 Softwarekonzeption der Satelliten-Systeme

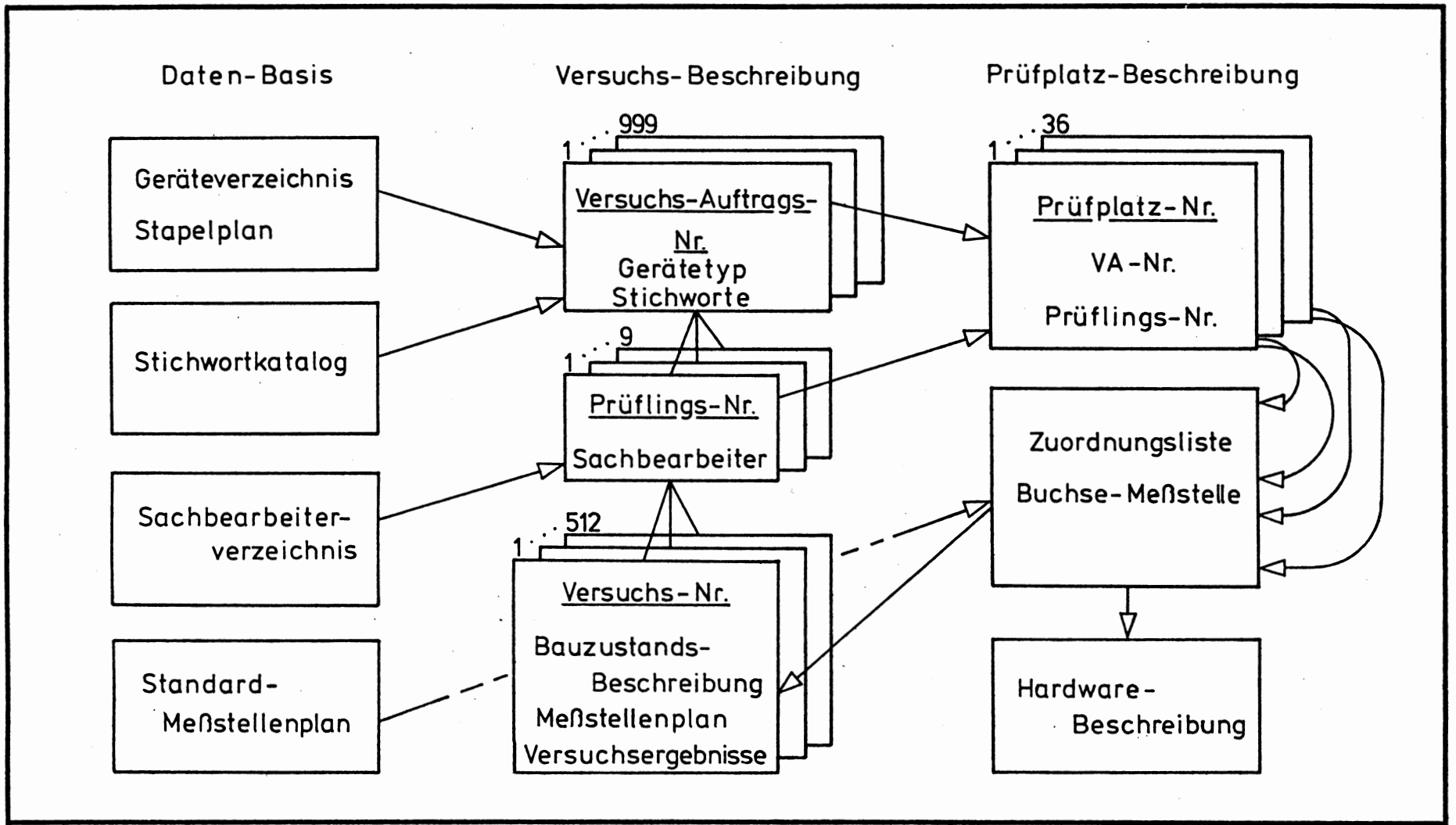


Abb.4 Dateien - Struktur

T. Martin (Hrsg.)

Industrielle Erfahrungen mit der Programmiersprache PEARL

Kernforschungszentrum Karlsruhe GmbH
PDV-Bericht, KfK-PDV 171, Juni 1979
50 Seiten, 13 Abbildungen, 5 Tabellen

Im Rahmen des Aussprachetages "Prozeßrechner" der VDI/VDE-Gesellschaft Meß- und Regelungstechnik fand am 28. März 1979 in Dortmund ein Erfahrungsbericht mit Diskussion zur Prozeßprogrammiersprache PEARL statt. Der Bericht enthält im Wortlaut vier Beiträge über die Anwendung von

PEARL auf den Gebieten

- Steuerung der Netzleitstelle eines Elektroversorgungsunternehmens,
- Steuerung eines verfahrenstechnischen Chargenprozesses,
- Steuerung des Materialflusses in einem flexiblen Fertigungssystem und der
- Automatisierung eines industriellen Entwicklungslabors,

sowie eine kurze Einführung in den Entwicklungsstand von PEARL und eine Zusammenfassung der Diskussion.

T. Martin (Hrsg.)

Industrielle Erfahrungen mit der Programmiersprache PEARL

Kernforschungszentrum Karlsruhe GmbH
PDV-Bericht, KfK-PDV 171, Juni 1979
50 Seiten, 13 Abbildungen, 5 Tabellen

Im Rahmen des Aussprachetages "Prozeßrechner" der VDI/VDE-Gesellschaft Meß- und Regelungstechnik fand am 28. März 1979 in Dortmund ein Erfahrungsbericht mit Diskussion zur Prozeßprogrammiersprache PEARL statt. Der Bericht enthält im Wortlaut vier Beiträge über die Anwendung von

PEARL auf den Gebieten

- Steuerung der Netzleitstelle eines Elektroversorgungsunternehmens,
- Steuerung eines verfahrenstechnischen Chargenprozesses,
- Steuerung des Materialflusses in einem flexiblen Fertigungssystem und der
- Automatisierung eines industriellen Entwicklungslabors,

sowie eine kurze Einführung in den Entwicklungsstand von PEARL und eine Zusammenfassung der Diskussion.

T. Martin (Ed.)

Industrial Experiences with the Realtime Programming Language PEARL

Kernforschungszentrum Karlsruhe GmbH
PDV-Report, KfK-PDV 171, June 1979
50 pages, 13 figures, 5 tables

The German Engineering Society for Measurement and Control Technique organized a panel discussion on "Industrial Experience with PEARL" on March 28, 1979, in Dortmund.

The report contains the four presentations given, dealing with application of PEARL in the areas of

- power distribution,
- chemical industry,
- flexible manufacturing systems, and
- laboratory automation.

The report also gives an overview of the status of PEARL and summarizes the discussion.

T. Martin (Ed.)

Industrial Experiences with the Realtime Programming Language PEARL

Kernforschungszentrum Karlsruhe GmbH
PDV-Report, KfK-PDV 171, June 1979
50 pages, 13 figures, 5 tables

The German Engineering Society for Measurement and Control Technique organized a panel discussion on "Industrial Experience with PEARL" on March 28, 1979, in Dortmund.

The report contains the four presentations given, dealing with application of PEARL in the areas of

- power distribution,
- chemical industry,
- flexible manufacturing systems, and
- laboratory automation.

The report also gives an overview of the status of PEARL and summarizes the discussion.