KfK-PDV 164
November 1978

PDV-Berichte

Untersuchung der Eignung der Prozeßrechnersprache PEARL zur Programmierung von Automatisierungsverfahren der zyklischen Prozeßdatenerfassung

R. Wiedenmann
Institut für Regelungstechnik und Prozeßautomatisierung
Universität Stuttgart

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.

Druck und Verbreitung:

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640 7500 Karlsruhe 1

Bundesrepublik Deutschland

KfK-PDV 164

PROJEKT PROZESSLENKUNG MIT DV-ANLAGEN

FORSCHUNGSBERICHT KfK-PDV 164

UNTERSUCHUNG DER EIGNUNG DER PROZESSRECHNERSPRACHE PEARL ZUR PROGRAMMIERUNG VON AUTOMATISIERUNGSVERFAHREN DER ZYKLISCHEN PROZESSDATENERFASSUNG

VON

WIEDENMANN, R.

INSTITUT FÜR REGELUNGSTECHNIK UND PROZESS-
AUTOMATISIERUNG, UNIVERSITÄT STUTTGART

131 SEITEN
31 ABBILDUNGEN
1 TABELLE
51 LITERATUR-
STELLEN

NOVEMBER 1978

Kurzfassung

Die Voraussetzungen der Sprachuntersuchung werden zusammengestellt und diskutiert. Es wird gezeigt, wie die Anforderungen von PEARL an ein Echtzeit-Betriebssystem durch die Anpassung des Betriebssystems des Prozeßrechners AEG 60-50 erfüllt werden.
Das zur Durchführung der Sprachuntersuchung gewählte gegenständliche Modell eines Fließprozesses mit der Aufgabenstellung Betriebsüberwachung wird dargestellt. Die charakteristischen Automatisierungsaufgaben zur Ermittlung der Sprachanforderungen werden an diesem Modell abgeleitet. Es wird diskutiert, wie die Anforderungen der zyklischen Prozeßdatenerfassung mit Sprachmitteln von PEARL gelöst werden können.

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Regelungstechnik und Prozeßautomatisierung der Universität Stuttgart.

Herrn Prof. Dr.-Ing. R. Lauber danke ich sehr herzlich für die Anregung zu dieser Arbeit sowie für Diskussionen und Hinweise bei ihrer Erstellung.

Herrn Prof. Dr. Ing. R. Quack danke ich für die Übernahme des Mitberichts.

Weiterhin gilt mein Dank der Projektleitung PDV in der Gesellschaft für Kernforschung mbH, daß diese Arbeit im Rahmen der Durchführung des Forschungsvorhabens "Prozeßprogrammiersystem auf der Grundlage von PEARL" entstehen konnte.

I N H A L T S V E R Z E I C H N I S

1. Einleitung

1.1 Problemstellung

Im Laufe der letzten Jahre wurden höhere Programmiersprachen für Prozeßrechner entwickelt, wie beispielsweise PAS1 [2] , RTL/2 [3] , PROCOL [4] und PEARL [5] .

Die Aufgabe, die sich beim Entwurf einer derartigen verfahrenorientierten Sprache stellt, ist die Entwicklung von Sprachelementen, mit denen die Lösungsverfahren der Prozeßautomatisierung programmiert werden können. Die Sprachmittel, die dabei zusätzlich zu denen bestehender verfahrenorientierter Sprachen erforderlich sind, lassen sich in zwei Gruppen einteilen:

1. Sprachmittel zur Behandlung von Prozeßdaten
2. Sprachmittel zur Organisation des zeitlichen Ablaufs von Programmen.

Eine der wesentlichen Forderungen an eine allgemeine Programmiersprache ist die Anwendungsbreite. Will man den überwiegenden Teil von Aufgabenstellungen der verschiedenen Prozeßtypen auf Sprachebene[+)] formulieren können, müssen die technischen Anforderungen (functional requirements) an derartige Sprachelemente bekannt sein und dem Sprachentwurf zugrunde gelegt werden.

Die Beurteilung der Sprachelemente, d.h. Bewertung ihrer Eigenschaften bzw. Fähigkeiten, erfolgt bisher aus Mangel an quantitativen Maßstäben durch subjektive Gewichtung der Spracheigenschaften:
Die Beurteilung setzt sich aus subjektiven, häufig sich widersprechenden Meinungen der Software-Technologen zusammen.
Die Meinungen der Betreffenden hängen ab von deren

- Erfahrungen mit Programmiersprachen
- fachlicher Qualifikation
- "Herkunft", z.B. ob der Betreffende Implementator oder Anwender ist.

[+)] ohne Einfügen rechnerabhängiger Prozeduraufrufe bzw. Betriebssystemeinsprünge

1.2 Ziel der Arbeit

Aufgrund der Tatsache, daß sowohl beim Entwurf als auch bei der Beurteilung der Sprachelemente für die genannten zusätzlichen Aufgaben nicht auf Erfahrungen von Sprachentwicklern zurückgegriffen werden kann - eben weil es noch keine Erfahrungen diesbezüglich gibt - wird hier der Weg gegangen, die Sprachuntersuchung anhand eines gegenständlichen Modells vorzunehmen:
Subjektive Meinungen sollen durch eine sachlich begründete Beurteilung ersetzt werden, indem die Sprachelemente den am Modellprozeß abgeleiteten Sprachanforderungen gegenübergestellt werden.

Die Darstellung der hierzu notwendigen Voraussetzungen soll der eigentlichen Sprachuntersuchung vorausgehen.

1.3 Gang der Arbeit

Die Sprachuntersuchung setzt einen technischen Prozeß voraus, wobei zwei mögliche Vorgehensweisen zu unterscheiden sind:

- Auswertung der Automatisierungsprogramme für einen industriellen Prozeß
- Auswertung der Programme zur Sprachuntersuchung mit Hilfe gegenständlicher Modelle, sog. Modellprozesse:
  Modellprozesse sind echte technische Prozesse, die in vereinfachter Form in einem Laboratorium aufgebaut sind.

Die Vor- und Nachteile beider Verfahren werden in [6] gegenübergestellt. Der Vorteil der zweiten Methode, nämlich die Möglichkeit echter Versuche zur Sprachuntersuchung, ohne zeitlichen Druck bzw. allzu großem Kostenaufwand, verbunden mit der experimentellen Ermittlung von Sprachanforderungen, führte zur Wahl dieses Weges.

Damit ergeben sich für diese Arbeit die folgenden Voraussetzungen:

1. PEARL-Kompiliersystem[+)]

   Zur Übersetzung und zum Ablauf der PEARL-Programme sind entsprechende Hilfsmittel notwendig:
   Zusätzlich zu einem Compiler und einem Test- und Bediensystem setzt eine Prozeßprogrammiersprache für den Ablauf gewisser Anweisungen ein geeignetes Echtzeit-Betriebssystem voraus.

+) Das PEARL-Kompiliersystem für den Prozeßrechner AEG 60-50 wurde in einer Gemeinschaftsarbeit der wissenschaftlichen Mitarbeiter A. Ghassemi, M. Helfert, A. Zeh und dem Autor im Rahmen der Arbeitsgemeinschaft ASME implementiert [7] - [10] .

Es muß die Funktionen enthalten, die von der Sprache gefordert werden, z.B. eine Zeitverwaltung für die Organisation der zyklischen Taskbeauftragung. Die Möglichkeiten und die gewählte Vorgehensweise für das Erreichen eines "PEARL-fähigen" Betriebssystems werden in Kapitel 2 aufgezeigt.

2. Modellprozeß "Betriebsüberwachung eines Fließprozesses"

Um vollständige Aussagen über die Eigenschaften einer Prozeßprogrammiersprache machen zu können, ist es erforderlich, die verschiedenen Sprachelemente an Prozessen unterschiedlichen Grundtyps zu untersuchen, also an einem Fließprozeß, Folgeprozeß und Stückprozeß [1].
Diese Forderung wurde durch den Aufbau mehrerer Modellprozesse am Institut für Regelungstechnik und Prozeßautomatisierung erfüllt. Die Sprachuntersuchungen lassen sich sinnvollerweise entsprechend dieser Grundtypen vornehmen, wobei dieser Arbeit ein Fließprozeß zugrundegelegt wird.
Neben der Festlegung des Prozeßtyps ist es erforderlich, mit ihm eine Aufgabenstellung zu verknüpfen. Als charakteristische Aufgabe wurde im vorliegenden Fall die Aufgabenstellung Betriebsüberwachung gewählt. Die Untersuchung am vorliegenden Modellprozeß (s. Kapitel 3) gewinnt dadurch an Bedeutung, daß bei dieser Aufgabenstellung und diesem Prozeßtyp der Prozeßrechnereinsatz am weitesten verbreitet ist. Es ist daher notwendig, die dort anfallenden Problemstellungen zu kennen und bei der Entwicklung bzw. Beurteilung einer Sprache zu berücksichtigen.

Das Hauptgewicht der Sprachuntersuchung sollte auf die Sprachmittel zur Formulierung prozeßrechnerspezifischer Aufgaben gelegt werden, also auf Sprachmittel zur Behandlung von Prozeßdaten bzw. zur Organisation des zeitlichen Ablaufs von Automatisierungsprogrammen.
Es sollten insbesondere diejenigen Sprachmittel untersucht werden, die charakteristisch für den jeweiligen Prozeßtyp und die zugehörige Aufgabenstellung sind. Im Falle der Betriebsüberwachung von Fließprozessen trifft dies vor allem auf die zyklische Programmbeauftragung zur Erfassung von Prozeßdaten zu.

Die Vorstellung der in PEARL dafür vorhandenen Sprachmittel sowie die Untersuchung der Eigenschaften dieser Sprachmittel wird in Kapitel 5 vorgenommen.

Zusammenfassend lassen sich die Bestandteile des nach [6] genannten "Prozeßrechner-Sprachlabors" einschließlich ihrem Zusammenwirken in Bild 1 folgendermaßen darstellen:

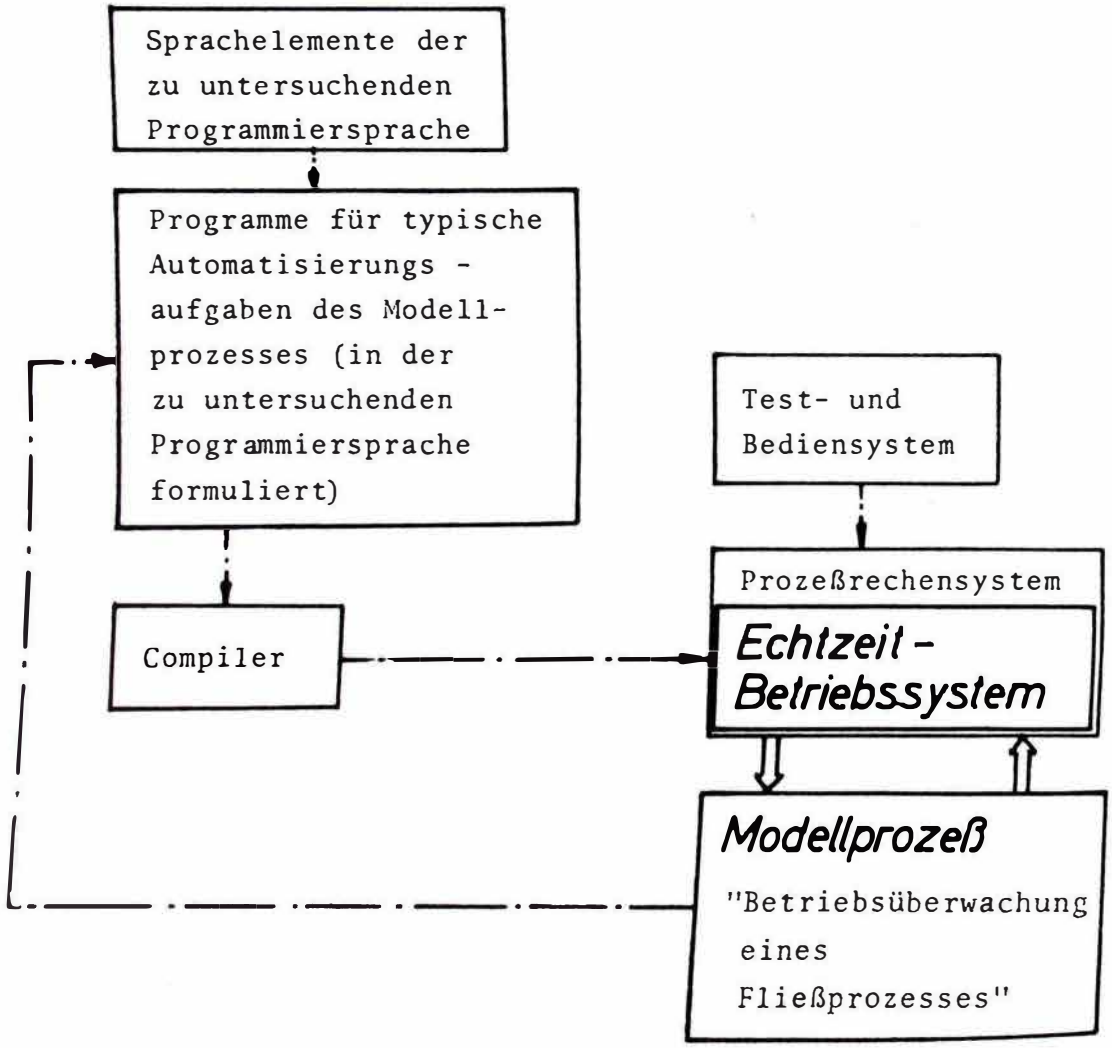


Bild 1: "Prozeßrechner-Sprachlabor" zur Sprachuntersuchung

2. Anpassung eines Betriebssystems nach den Anforderungen von PEARL

2.1 Begründung der Notwendigkeit einer Anpassung

Ziel der Entwicklung von PEARL war es, eine anwenderfreundliche Programmiersprache für den Ingenieur zu erstellen.
Kriterien dafür waren in erster Linie

- Leichte Erlernbarkeit
- Einfachheit der Formulierung von Automatisierungsaufgaben
- Zuverlässigkeit der Programme
- Dokumentationsfreundlichkeit
  und leichte Lesbarkeit der Programme.

Aufgrund dieser Tatsache konnte die Implementationsfreundlichkeit des PEARL - Kompiliersystems nicht in besonderem Maße berücksichtigt werden, insbesondere nicht die maschinenabhängigen Fähigkeiten bestehender, firmenspezifischer Echtzeit-Betriebssysteme.
Die Anforderungen von PEARL an Betriebssysteme rühren von Sprachanweisungen her, die direkt Betriebssystemaufrufe (supervisor calls) zur Folge haben, also im wesentlichen von

- Anweisungen für die parallele Task-Ablaufsteuerung[+)]
- Anweisungen für die Interruptbehandlung
- Ein/Ausgabeanweisungen.

Diese Anforderungen sind dabei zum großen Teil durch die Semantik der dazu notwendigen Sprachelemente eindeutig definiert. PEARL fordert von einem Betriebssystem das Vorhandensein der Bausteine (Module) zur Durchführung dieser Anweisungen (s. Beispiele in Bild 2.1 und 2.4). Abhängig vom Zielrechner-Betriebssystem wird ein Teil dieser Bausteine in unveränderter Form übernommen werden könnnen, im allgemeinen wird es aber notwendig sein, mit mehr oder weniger großem Aufwand ein "PEARL-fähiges" Betriebssystem zu erzeugen. Dieses ist die erste Voraussetzung für den Lauf übersetzter PEARL-Programme und damit für die Sprachuntersuchung. Aus diesem Grund sollen die entsprechenden Lösungsmöglichkeiten und eine Beschreibung der gewählten Lösung folgen.

[+)] Unter einer Task ist dabei die Durchführung einer Automatisierungsaufgabe durch ein Programm zu verstehen, wobei der Ablauf von einem Betriebssystem gesteuert wird.

Bedeutung	PEARL	Abgesetzte Assemblerbefehle (AEG 60-50)
Start einer Task mit dem Namen T1	ACTIVATE T1;	LDA TASKNUMMER (Tasknummer von T1 in den Akkumulator laden) SPB P&ACT (Unterprogrammsprung in eine Betriebssystemroutine mit dem Namen P&ACT)
Sperren des Interrupts I1	DISABLE I1;	LDA ITRNR (Interruptnummer von I1 in den Akkumulator laden) SPB P&DISA (Unterprogrammsprung in eine Betriebssystemroutine mit dem Namen P&DISA)

Bild 2.1: Beispiele für Betriebssystemaufrufe

2.2 Lösungsmöglichkeiten zur Erfüllung der Betriebssystem-Anforderungen von PEARL

Setzt man voraus, daß ein Echtzeit-Betriebssystem zur Verfügung steht, das noch nicht den Anforderungen von PEARL genügt, so gibt es drei Verfahren zur Erzeugung eines "PEARL - fähigen" Betriebssystems. Für eine Auswahl eines Verfahrens ist die Kenntnis der Vor- und Nachteile erforderlich. Sie werden in den folgenden Abschnitten vorgestellt.

2.2.1 Erstellung eines neuen Betriebssystems

Die erste Alternative ist die Erstellung eines neuen Betriebssystems entsprechend den speziellen Anforderungen von PEARL.

Nach [11] lassen sich die Programmbausteine eines Betriebssystems in drei Bereiche gliedern:

1. Programmbausteine zur Durchführung des regulären Betriebsablaufes
2. Programmbausteine zur Durchführung des irregulären Betriebes, z.B. Wiederanlauf nach Störungen
3. Programmbausteine zur Vorbereitung und Unterstützung des Betriebsablaufes.

Je nach den Benutzer-Anforderungen bzw. den vorhandenen Mitteln lassen sich zwei verschiedene Varianten definieren, welche die Neuerstellung aller bzw. nur einzelner Bereiche erfordern:

1. Das neue Betriebssystem umfaßt alle Programmbausteine von Bereich 1 und Bereich 2.
   Von Bereich 3 sind nur diejenigen Teile enthalten, die für den Ablauf und den Test der Anwenderprogramme notwendig sind, also Testprogramme und Standardfunktionen.

   Die Nachteile dieser Lösung sind:

   - Beträchtlicher Aufwand zur Neuerstellung der erwähnten Bausteine
   - Keine Möglichkeit einer Programmübersetzung und des Laufs vorhandener Dienst- bzw. Assembler-Anwenderprogramme parallel zu PEARL-Anwenderprogrammen:
     Für das Übersetzen von PEARL-Moduln sowie den Lauf von Assembler-Dienst- und Anwenderprogrammen muß jedesmal ein Betriebssystemwechsel zum ursprünglichen Betriebssystem erfolgen.

2. Das neue Betriebssystem umfaßt alle Bausteine der drei Bereiche.
   In diesem Fall ist eine Programmübersetzung bzw. der Lauf von Dienstprogrammen parallel zum Ablauf von PEARL-Anwenderprogrammen möglich, genügender Speicherplatz vorausgesetzt.

   Nachteilig ist hier der sehr große Aufwand zur Neuerstellung aller Betriebssystembausteine, insbesondere für die i.a. sehr zahlreichen und umfangreichen Dienstprogramme.

Vorteilhaft bei einer Betriebssystem-Neuerstellung ist die erreichbare Effektivität der Betriebssystem-Bausteine hinsichtlich Speicherplatz und Laufzeit, da die Anforderungen von PEARL bereits bei der Konzeption des Betriebssystems berücksichtigt werden können, einschließlich dem Ausnutzen der Hardwarestruktur des Rechners.

2.2.2 Implementierung eines universellen PEARL-Betriebssystems

Im Rahmen des Projektes PDV der Gesellschaft für Kernforschung wurde ein universelles PEARL-Betriebssystem entwickelt [12] .

Vorteile:

- Teilweise Maschinenunabhängigkeit
- Der Implementationsaufwand gegenüber einem speziellen, rechnerabhängigen Betriebssystem wird auf ein Drittel reduziert (ohne Gerätetreiber und Programme für die formatierte Ein/Ausgabe)
- Leichte Änderbarkeit bzw. Erweiterbarkeit.

Nachteile:

- Der Sprachumfang, der diesem Betriebssystem zugrunde gelegt wurde, entspricht nicht dem aktuellen Stand von PEARL
- Durch die Maschinenunabhängigkeit können spezielle Hardwarestrukturen von Rechnern nicht bzw. nur nach Änderungen des universellen PEARL-Betriebssystems genutzt werden.
  Beispiel (siehe auch [12] Anhang B23):
  Retten von Registerinhalten in
  - Keller (moderne Rechner)
  - "Prozeßkontrollsatz" im Hauptspeicher (PEARL-Betriebssystem).
- Das Betriebssystem ist in der PL/1-ähnlichen Sprache GBL1 geschrieben dadurch wird die Erstellung eines maschinenabhängigen Übersetzers GBL1 - Assemblercode notwendig.
- Das Betriebssystem ist bisher nur durch Simulation auf einem Großrechner ausgetestet, es gibt keine praktischen Erfahrungen mit Prozeßrechnern.
- Nachteile von Abschnitt 2.2.1 Punkt 1.

2.2.3 Anpassung eines bestehenden Betriebssystems

Neben den bereits aufgeführten Verfahren bietet sich die Anpassung eines bestehenden Betriebssystems an.

Vorteile:

- Es müssen nur diejenigen Betriebssystem-Module angepaßt werden, die aufgrund der Anforderungen von PEARL geändert werden müssen, die übrigen Teile können unverändert bleiben, d.h. der Aufwand bleibt beschränkt.
- Bestehende Programmbausteine zur Vorbereitung und Unterstützung des Betriebsablaufs bedürfen keiner Modifikation.
- Bisherige Anwenderprogramme können unverändert ablaufen, im allgemeinen auch quasi-parallel zu PEARL-Tasks.

Nachteile:

- Es ist die genaue Kenntnis des bestehenden Betriebssystems notwendig.
- Die Erweiterungen führen evtl. zu einer "Doppelverwaltung", welche sowohl die Laufzeit als auch den Speicherplatz der zusätzlichen Module betrifft.

Beispiel (AEG 60-50): Programmidentifikation und Priorität

Assembler- bzw. FORTRAN-Programme: Die Identifikation und Priorität sind in der sog. Programmnummer (PNR) zusammengefaßt; je größer diese Nummer ist, desto höher ist die Priorität.

PEARL: Die Identifikation erfolgt durch eine Tasknummer, die vom Compiler festgelegt wird, sowie der PNR: Doppelidentifikation. Priorität: PNR und Prioritätszahl.

- Einschränkungen bei der Erweiterung eines bestehenden Betriebssystems infolge nur schwer überwindbarer Schwierigkeiten können zu Implementationsabhängigkeiten führen.
  Beispiel: Die kleinste Zykluszeit beträgt bei Programmen der AEG 60-50 aufgrund vorhandener Zeitverwaltungsfunktionen 1 Sekunde.

2.3 Beschreibung des gewählten Lösungsweges

Ziel der Implementierung des ASME - PEARL - Subsets Stufe 1 (APS1) war die praktische Erprobung der Sprachelemente von PEARL mit der Randbedingung einer möglichst raschen Erstellung des gesamten Kompiliersystems. Das bedeutete, daß der Aufwand der Betriebssystem-Adaption auf notwendige Erweiterungen beschränkt werden mußte. Der Schwerpunkt der Tätigkeit lag auf der Sprachuntersuchung, nicht auf der Implementierung möglichst effektiver Betriebssystemroutinen. Aus diesen Gründen wurde das Verfahren der Anpassung des bestehenden Betriebssystems MARVIS (multi-access Realzeitverwaltung durch Interrupt Steuerung) ausgewählt. In diesem Abschnitt sollen die wesentlichen Merkmale des Lösungsweges aufgezeigt werden.

2.3.1 Übersicht über die Ablauforganisation von MARVIS

Stellvertretend für die Teilaufgaben der Betriebssystemanpassung, nämlich Adaption von

- Ablauforganisation
- Interrupt-Zeitverwaltung
- Standard-Ein/Ausgabe
- Ein/Ausgabe mit peripheren Speichern
- Prozeßsignal-Ein/Ausgabe

wird hier diejenige der Ablauforganisation vorgestellt. Aus der Gegenüberstellung der vorhandenen Ablauforganisation und den PEARL-Anforderungen sollen Aufwand und Vorgehen bei einer solchen Adaption verdeutlicht werden.
Voraussetzung einer erfolgreichen Erweiterung ist die genaue Betriebssystemkenntnis:
Die erste Aufgabe war daher die Zusammenstellung dessen Fähigkeiten und Aufbau, um einen Überblick über den "Betriebssystem-Istzustand" zu gewinnen. Dabei ergibt sich die folgende Übersicht [13] :

Aufgaben der MARVIS - Ablauforganisation

Bezüglich der Organisationsaufgaben, die gelöst werden müssen, erfolgt eine Aufteilung in zwei Verwaltungen:

- Auftragsverwaltung

  In dieser Verwaltung werden Aufträge an Tasks notiert, wobei andere Tasks oder Binärsignaländerungen die Auftraggeber sein können.

- Aufrufverwaltung

  Hier erfolgt die Abarbeitung von erteilten Aufträgen bis zum Start der Tasks einschließlich der Beendigung der Aufträge.

Die Entkopplung beider Verwaltungen ist in Bild 2.2 ersichtlich.

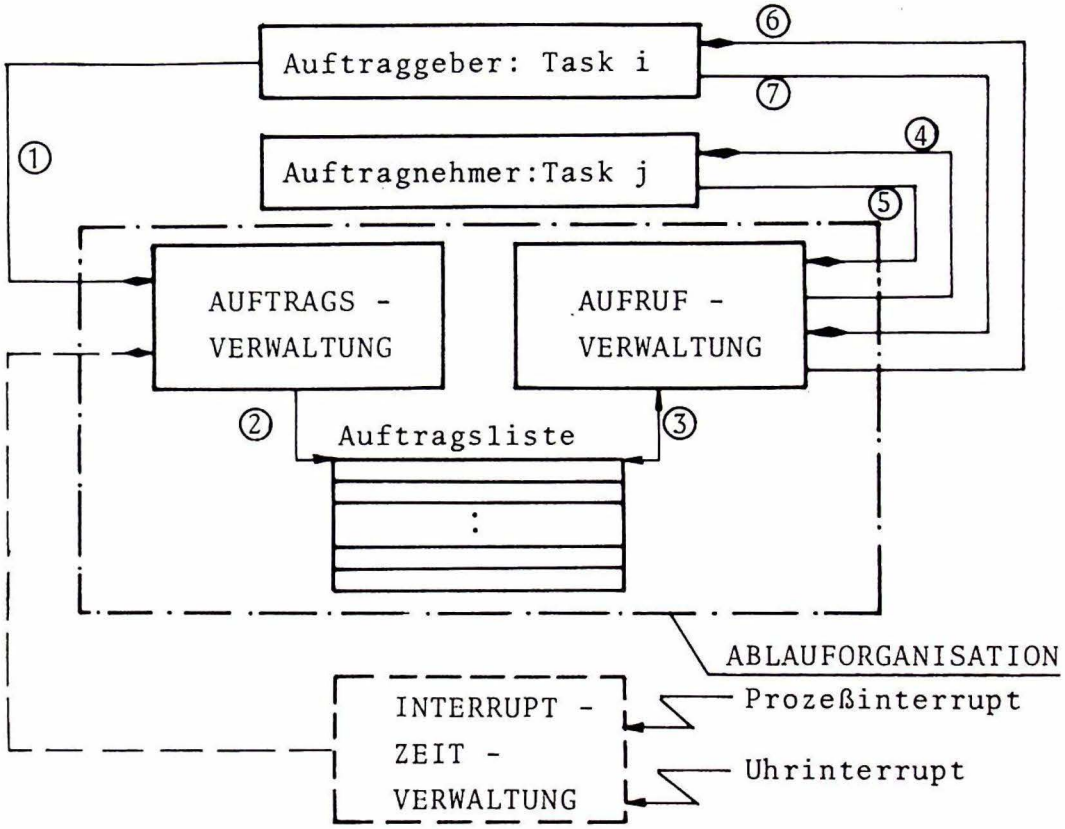


Bild 2.2 : Ablauforganisation von MARVIS

Der Auftraggeber - hier Task i - übermittelt der Auftragsverwaltung einen Auftrag (1): Aktivierung der Task j, d.h. Eintragung in die Auftragsliste (2). Der Auftrag wird von der Aufrufverwaltung entnommen (3), der Auftragnehmer j wird gestartet (4)[+)]. Nach Ablauf der Task j erfolgt eine Rückmeldung (5) an die Ablauforganisation, anschließend wird Task i fortgesetzt (6). Nach Beendigung dieser Task wird in die Aufrufverwaltung zurückgekehrt (7).

[+)] Die Task j sei wichtiger als die Task i

Zu erwähnen ist hier, daß die in Bild 2.2 eingezeichnete Interrupt- bzw. Zeitverwaltung sowohl bei MARVIS als auch bei der anschließend beschriebenen PEARL - Erweiterung nicht behandelt wird.

Auftragsverwaltung

MARVIS unterscheidet zwischen drei Beauftragungsarten:

- 1 Aktivierung von Tasks ohne Parameterübergabe
- 2 Aktivierung von Tasks mit Parameterübergabe über einen generellen, verketteten Auftragspuffer zur Parameter-Zwischenspeicherung [14].
  Spezielle Betriebssystemroutinen übernehmen die Einspeicherung und Entnahme der Parameter, wobei für jede Task der maximale Pufferbedarf angegeben werden muß.
  Bei vollem Puffer werden pufferbelegende Tasks zurückgestellt.
- 3 Aktivierung von Tasks mit Übergabe der Auftraggeberidentifikatio
  Diese Beauftragungsart zum Anstoß dringlicher, residenter Tasks wurde eingeführt, um auch bei einem eventuellen Pufferüberlauf (siehe -2)eine Auftragsverzögerung zu verhindern:
  Einführung mehrerer nummerierter"Ereigniskeller".

Aufrufverwaltung

Diese Verwaltung bringt beauftragte Tasks zum Ablauf.
Liegen keine Aufträge vor, läuft sie in einer "Warteschleife" (Prozessor-Leerlauf). Eine Notierung in den Auftragslisten führt dazu, daß die Verwaltung die Warteschleife verläßt und die be - auftragte Task startet. Nach deren Beendigung wird in diese Verwaltung zurückgekehrt.
Im einzelnen werden folgende Aufgaben ausgeführt:

- 1 Suchen und Entnehmen von Aufträgen aus den Auftragslisten, Bestimmung von Auftragnehmer und evtl. Auftraggeber
- 2 Untersuchung der Lauffähigkeit beauftragter Tasks:

Zurückstellungen können erfolgen durch:

- Pufferüberlauf: Pufferschreibzeiger > Maximalwert (s. Auftragsverwaltung, Punkt - 2); eine pufferbelegende Task wird zurückgestellt und zwar so lange, bis wieder Platz im Puffer vorhanden ist.
- Synchronisation: hier gegenseitiger Ausschluß von Tasks

- 3 Untersuchung des Task-Typs, d.h. der Prioritätsverhältnisse und Speicherresidenz von Tasks

- 4 Untersuchung, ob sich die beauftragte Task bereits im Hauptspeicher befindet:
  - wenn nein: Externspeicherverwaltung beauftragen
  - wenn ja: kein Transfer notwendig

- 5 Untersuchung, ob für die beauftragte Task genügend Platz im Hauptspeicher ist:
  - wenn nein: Zurückstellung des Auftrags
  - wenn ja: Suchen und Belegen eines geeigneten Speicherbereichs

- 6 Bestimmung der dringlichsten, lauffähigen Task (Prozessorvergabe)

- 7 Freigabe des belegten Speicherbereichs nach erfolgtem Lauf einer Task.

Programmorganisation, Prozessorvergabe

Die Aufrufverwaltung beginnt ihre Untersuchung nach Aufträgen in der Reihenfolge der Priorität, wobei Priorität identisch ist mit Programmnummer und Programmidentifikation.
Grundsätzlich unterscheidet man zwischen drei Prioritätsebenen, wobei dann eine beauftragte Task einer höheren Ebene eine laufende niederpriore Task unterbricht. Aufträge innerhalb einer Ebene dagegen werden nur vorgemerkt und am Ende einer laufenden Task entsprechend dem Prioritätsprinzip abgearbeitet. Die Anzahl der Ebenen wurde aus folgendem Grund auf drei festgelegt:
Um Rücktransfers und damit auch eventuelle Holtransfers von unterbrochenen Tasks zu vermeiden, bleiben diese im Hauptspeicher liegen. Eine Ausnahme bilden Hintergrundprogramme, die bei Bedarf ausgeräumt werden.

Eine größere Anzahl von Ebenen würde dementsprechend die Möglichkeit der Hauptspeicherblockierung durch unterbrochene Tasks vergrößern, während eine noch kleinere Anzahl die Wartezeit lauffähiger Tasks verlängern würde.

Die Programme werden in drei Ebenen, auch Klassen genannt, aufgeteilt (Bild 2.3).

Programmtyp A[+)]

Residente Programme, die zur schnellen Auswertung von Prozeßsignaländerungen eingesetzt werden. Sie sollen eine kurze Laufzeit habe' da sie sich von keinem anderen Programmtyp unterbrechen lassen.

Programmtyp B[++)]

Programme für den Realzeitbetrieb (DIN 44300) mit einer maximalen Laufzeit von 100 msec...200 msec. Sie liegen i.a. auf dem Hintergrundspeicher, eine gewisse Anzahl kann auch resident gehalten werden.

Programmtyp C[+++)]

Programme, die nicht direkt an den Realzeitbetrieb gebunden sind (Hintergrundprogramme). Sie können bei Bedarf - aber nur vom Programmtyp C veranlaßt - aus dem Hauptspeicher ausgeräumt werden. Im Gegensatz zur Verwaltung der oben erwähnten Programme liegt diejenige dieses Typs nahezu vollständig auf dem Hintergrundspeicher und muß vor einem Programmlauf in den Hauptspeicher gebracht werden, womit natürlich eine große Verwaltungslaufzeit verbunden ist. Diese Klasse kann zusätzlich noch in maximal acht Unterklassen (hier drei) mit unterschiedlichen Prioritäten unterteilt werden.

+) Herstellername: Kernspeicherprogramme
++) Prozeßprogramme
+++) Komplexprogramme

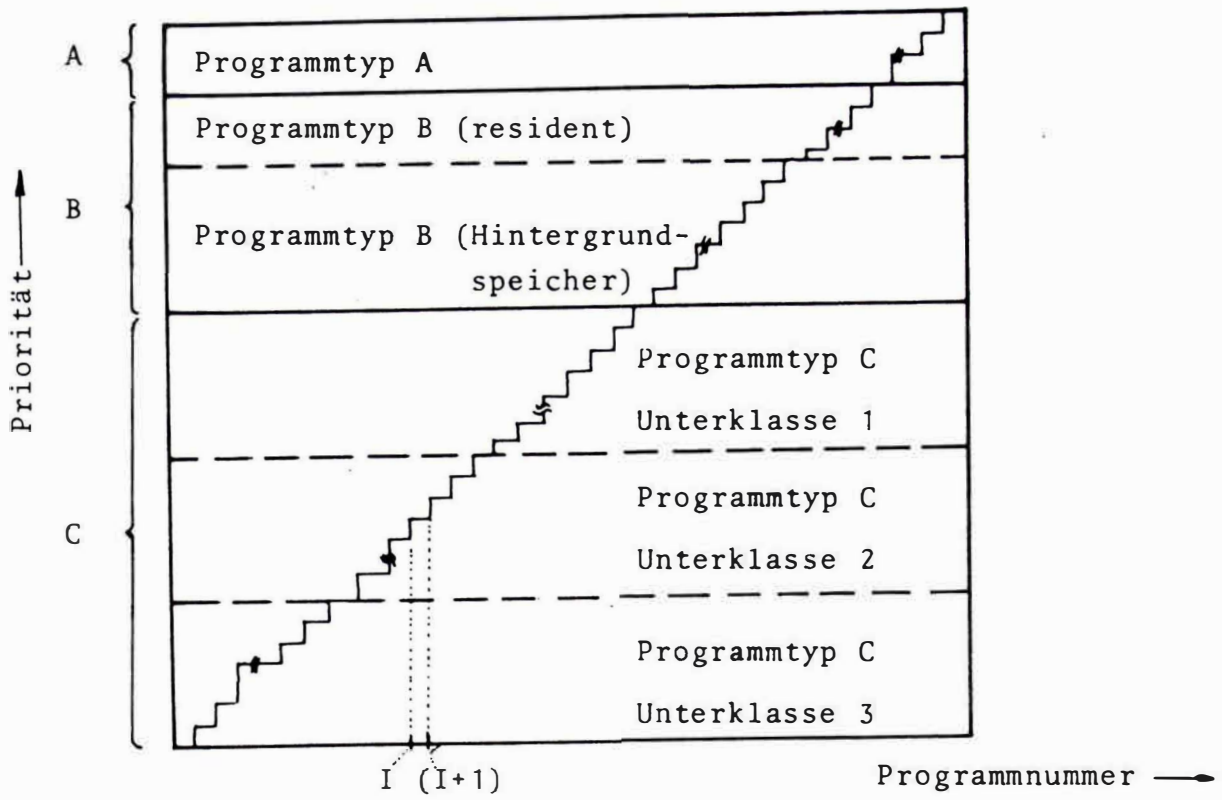


Bild 2.3: MARVIS - Programmorganisation

Synchronisation

Die Synchronisation von Tasks kann beim sog. Programmdateneinbau+) über die Korrespondenz statisch mit Hilfe von Verriegelungs-bitmustern vorgenommen werden.
Tasks mit demselben Muster dürfen nicht (quasi-) parallel ablaufen ("gegenseitiger Ausschluß").

+) Eintragung charakteristischer Programmdaten in Betriebssystemlisten (z.B. Programmlänge, Startadresse usw.)

2.3.2 Anforderungen von PEARL an die Ablauforganisation eines Betriebssystems

PEARL fordert das Vorliegen der Betriebssystem-Module zur Realisierung der parallelen Task-Ablaufsteuerung bzw. Task-Koordinierung. Die entsprechenden Anweisungen und ihre Bedeutung sind in der nachstehenden Tabelle (Bild 2. 4) zusammengefaßt[+].

Die dabei verwendeten Symbole haben folgende Bedeutung:

[ ] Option, d.h. Angabe nicht obligatorisch

{ } Zusammenfassung für konjunktiv oder disjunktiv verknüpfte Elemente.

Anweisung	Bedeutung
ACTIVATE taskname [PRIORITY zahl] ;	Start der Ausführung einer Task (Beauftragung). Durch die statische Priorität wird die Wichtigkeit einer Task im Rechensys angegeben.
SUSPEND [taskname] ;	Unterbrechen der Ausführung ein Task. Die Angabe des Tasknamens kann entfallen, wenn die Task, unter deren Kontrolle die Anweisung ausgeführt wird, unterbrochen werden soll.
CONTINUE taskname;	Fortsetzen einer durch SUSPEND angehaltenen Task.
{Zeit-Interrupt Bedingung} RESUME; (entspricht einem sofortigen SUSPEND und einem bedingten Fortsetzen mit CONTINUE)	Unterbrechen des Ablaufs einer Task für eine bestimmte Zeitspanne bzw. bis zum Eintreffen eines Interrupts.

+) Dieser Tabelle wird die APS1-Implementierung zugrunde gelegt.

Fortsetzung der Tabelle

Anweisung	Bedeutung
TERMINATE [taskname] ;	Beendigung der Ausführung einer Task (Bedeutung von taskname siehe SUSPEND)
REQUEST Semaphorebezeichner;	Mit der REQUEST-Anweisung kann die Fortsetzung einer Task vom Wert einer Semaphore-Variablen [16] abhängig gemacht werden.
RELEASE Semaphorebezeichner;	Diese Anweisung bewirkt die Erhöhung einer Semaphore-Variablen um 1. Damit kann evtl. die durch eine vorherige REQUEST-Anweisung blockierte Task wieder fortgesetzt werden.

Bild 2.4: Task-Steuer- bzw. Synchronisieranweisungen des APS1

Im Zusammenhang mit diesen Anweisungen nehmen Tasks bei ihrem Ablauf verschiedene Taskzustände an, denen einschließlich ihrer Übergänge eine große Bedeutung bei der Betriebssystemanpassung zukommt. Die in [1] und [15] eingeführten Begriffe "ruhend", "blockiert", "bereit" und "laufend" werden als bekannt vorausgesetzt und übernommen. In Erweiterung zu [15] soll der Zustand "blockiert" aufgeteilt werden und zwar aus zweierlei Gründen:

1. Unterscheidung der Ursache der Blockierung bzw. der Fortsetzungsbedingungen
2. Unterscheidung bzgl. der Betriebsmittelbelegung:
   Die Betriebsmittelverwaltung ist stark von der Art der Blockier-Anweisung abhängig.

Insgesamt lassen sich vier verschiedene "blockiert" - Zustände definieren:

- "blockiert1" : Bei diesem Zustand erfolgte eine Blockierung durch die Taskoperation SUSPEND bzw. RESUME ;
- "blockiert2" : Dieser Zustand wird durch die Anweisung REQUEST S(S=0) erreicht
- "doppeltbehindert": Befindet sich eine Task im Zustand "blockiert2", so kann sie durch eine auf sie angewandte SUSPEND - Anweisung zusätzlich blockiert, also "doppeltbehindert" werden. Ihre Fortsetzung erfolgt nur, wenn sie über eine CONTINUE- und eine Synchronisier - anweisung RELEASE S (S=0) in den Zustand "bereit" übergeführt wird [17] .
- "E/A-unterbrochen" : Dieser betriebssysteminterne Zustand wird erreicht, wenn eine Task bei einer Ein/Ausgabe - Anweisung den Prozessor freigibt und dann auf die Gerätefertigmeldung wartet. Je nach Ein- bzw. Ausgabezeit wird der Task das Betriebsmittel Hauptspeicher entzogen, d.h. nach Eintreffen des Gerätefertiginterrupts muß die Task evtl. wieder in den Hauptspeicher transferiert werden.
  Beispiel:
  "schnelles" Gerät: Analogwerteingabe bei Relaisadreßumschaltung (ca. 5 msec): keine Speicherplatzfreigabe
  "langsames" Gerät: Teletype; die Task muß ihren belegten Speicherplatz freigeben.

Mit dieser Darstellung lassen sich drei prinzipiell verschiedene Arten von Taskzustandsübergängen angeben, die von der Taskverwaltung auch mit sehr unterschiedlichem Aufwand realisiert werden müssen:

- Zustandsübergänge, hervorgerufen durch Task - Steueranweisungen: "beabsichtigte" Steuerung der Übergänge durch den Benutzer
- Zustandsübergänge, hervorgerufen durch Task - Synchronisieranweisungen:
  Die Übergänge sind abhängig vom augenblicklichen Zustand des Synchronisier-Mittels (hier Semaphore-Variable).
- Zustandsübergänge, hervorgerufen durch betriebssysteminterne Maßnahmen, z.B. "laufend" nach "bereit" oder "laufend" nach "E/A-unterbrochen":
  kein Einfluß des Benutzers.

Diese Übergänge lassen sich in einem Task - Zustandsdiagramm darstellen (s. Bild 2. 5).

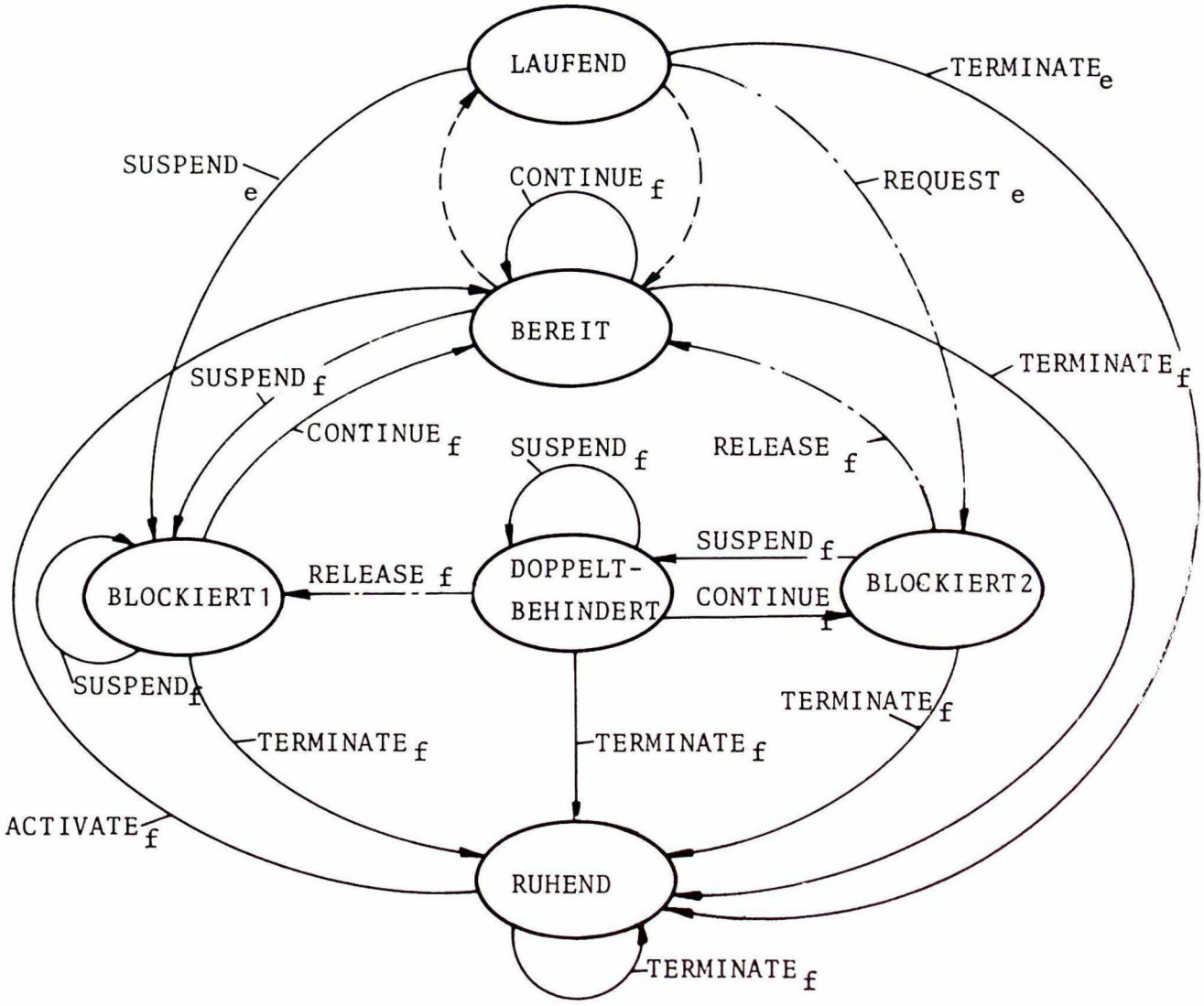


Bild 2. 5: Taskzustandsübergänge des APS1

In Bild 2.5 bedeuten:

——————— Übergänge, veranlaßt durch Task - Steueranweisungen
—·—·—·— Übergänge, veranlaßt durch Task - Synchronisieranweisungen
— — — — — Übergänge, veranlaßt durch das Betriebssystem (intern)

Index e Übergänge, welche den Zustand der laufenden Task betreffen (eigen)

Index f Übergänge, welche den Zustand anderer Tasks betreffen (fremd

2.3.3 Notwendige Erweiterungen der bestehenden Ablauforganisation

Aus der Beschreibung der Ablauforganisation von MARVIS und den Anforderungen von PEARL ergibt sich folgende Aufstellung:

Anweisungen	Vorhanden (x) bzw. fehlt (-) in MARVIS
ACTIVATE	x
SUSPEND	-
CONTINUE	-
RESUME	-
TERMINATE	
- eigene Task	x
- fremde Task	-
REQUEST	-
RELEASE	-

Aus dieser Aufstellung ist ersichtlich, welche Anweisungen mit der bestehenden Ablauforganisation prinzipiell realisiert werden konnten und welche Betriebssystembausteine vollkommen neu erstellt werden mußten.

2.3.4 Lösungsmöglichkeiten zur Realisierung der Anpassung der bestehenden Ablauforganisation

Eine erste, sehr wesentliche Aufgabe vor Beginn der Adaption war die Untersuchung der verschiedenen Lösungsmöglichkeiten mit einer Abschätzung von

- Zeitaufwand
- Effektivität
  - Verwaltungslaufzeit (und damit auch Laufzeit von Tasks)
  - zusätzlicher Speicherplatz.

Neben diesen Aspekten war außerdem

- die Auswirkung der Implementierung auf Spracheigenschaften (Implementationsabhängigkeiten)
- die Rückwirkung auf den Ablauf vorhandener Organisations- bzw. Automatisierungsprogramme

zu untersuchen.

Eine Gegenüberstellung der verschiedenen Varianten nach obigen Gesichtspunkten hatte dann das Ziel, die günstigste Lösung zu finden und zu implementieren.
Vorgenommen werden konnte diese Untersuchung im wesentlichen anhand von

- Betriebssystembeschreibungen
- Intensivem Meinungsaustausch mit Betriebssystemspezialisten des Rechnerherstellers
- Test des Ablaufs einzelner Betriebssystem-Module.

Die zentrale Frage, die sich bei der Adaption der MARVIS - Ablauforganisation stellte, war diejenige der Zuordnung von PEARL - Tasks zu Prioritätsebenen im Sinne des Programmtyps von MARVIS. Da Tasks mit unterschiedlicher Priorität dort sehr unterschiedlich verwaltet werden (die Verwaltungen sind teils resident, teils nichtresident), war es aus Gründen des Organisationsaufwandes nicht möglich, PEARL - Tasks in mehrere solcher Ebenen im Sinne von MARVIS einzuordnen. Infolge dieser Tatsache mußten sie einer Klasse, d.h. derjenigen von Programmtyp B oder Programmtyp C zugeordnet werden. Die Vor- und Nachteile beider Lösungen werden in der folgenden Gegenüberstellung aufgeführt:

Lösung des Programmtyps B

Diese Klasse wird in mehrere Unterklassen mit verschiedenen, vom Anwender angebbaren Prioritäten aufgeteilt. Um ein Blockieren des Hauptspeichers zu vermeiden, muß logischerweise bei Bedarf ein Ausräumen unterbrochener bzw. blockierter Tasks ermöglicht werden.
Vorteilhaft sind die Aufteilung in "beliebig" viele Ebenen sowie die geringen Verwaltungslaufzeiten, da die gesamte Verwaltung resident gehalten werden kann.
Als Nachteil ergibt sich der hohe Änderungsaufwand und viel zusätzlicher Speicherplatz.

Lösung des Programmtyps C

Diese Lösung sieht maximal drei Prioritätsebenen vor, wobei jede Ebene einer der vorhandenen Unterklassen des Programmtyps C angehört.
Der Vorteil liegt im geringen Änderungsaufwand bzw. wenig zusätzlichem Speicherplatz, da bereits drei Unterklassen realisiert sind.
Nachteilig sind drei Punkte:
- Sehr hohe Verwaltungslaufzeiten, da die Verwaltung nur teilweise resident ist
- Durch die Beschränkung auf drei Prioritätsebenen ergibt sich eine starke Implementationsabhängigkeit.
- Es treten bei quasi-parallelem Ablauf von vorhandenen Organisationsprogrammen bzw. Anwenderprogrammen des Typs C (Hintergrundprogramme langer Laufzeit) sowie von laufbereiten PEARL-Tasks evtl. sehr hohe Wartezeiten dieser Tasks auf.

Die Nachteile des Lösungsweges Programmtyp C, also hohe Verwaltungslaufzeiten und Beschränkung auf drei Prioritätsebenen, führte zur Wahl des Lösungsweges 'Programmtyp B'. Teile der Implementierung werden im nächsten Abschnitt vorgestellt.

2.3.5 Übersicht über das Gesamtkonzept der gewählten Lösung

Es werden nun die wichtigsten Merkmale der Tasking-Implementierung auf der AEG 60-50 vorgestellt. Zu erwähnen ist an dieser Stelle, daß sich diese PEARL-Implementierung auf ein Ein-Modul-Konzept mit maximal 48 Tasks bezieht. Es kann also immer nur ein PEARL-Modul geladen d.h. im Betriebssystem eingetragen sein.

-1 Alle PEARL-Tasks werden dem Programmtyp B zugeordnet. Ihre Programmnummern liegen in einem definierten Bereich. Das Betriebssystem erkennt daran, daß es sich um PEARL-Tasks handelt.

Ferner ist gewährleistet, daß ASSEMBLER- und FORTRAN-Programme - auch quasiparallel zu PEARL-Tasks - unter dem bisherigen Prioritätsschema laufen können.

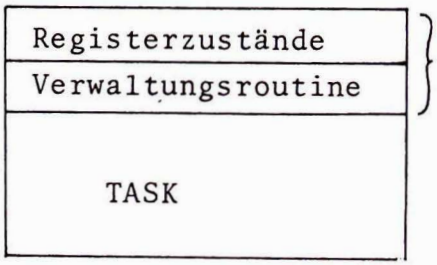
- 2 Die Taskidentifikation erfolgt über Tasknummern, die zur Kompilierzeit festgelegt werden. Zu bemerken ist, daß sie im Gegensatz zu den Programmnummern nicht im Zusammenhang mit der Taskpriorität stehen.
Die Zuordnung Tasknummer - Programmnummer erfolgt nach dem Übersetzen entsprechend der Priorität.

- 3 Eine Task, die von einer höherprioren Task unterbrochen wird, bleibt so lange wie möglich laufbereit im Hauptspeicher liegen.

- 4 Tasks, die infolge der Anweisungen SUSPEND und/oder REQUEST am Weiterlauf gehindert werden, bleiben ebenfalls so lange wie möglich im Hauptspeicher. In einer zweiten Betriebssystemversion werden sie dagegen sofort nach dem Blockieren aus dem Hauptspeicher verdrängt.

- 5 Ist eine beauftragte Task zur Zeit des Aufrufs noch nicht im Hauptspeicher, so läuft die laufbereite Task höchster Priorität weiter , aber höchstens so lange, bis der Holtransfer für die beauftragte Task beendet ist.

- 6 Am Beginn jeder Task wird vom Compiler für die Abspeicherung der aktuellen Registerzustände bei einer Programmunterbrechung sowie für eine kurze Verwaltungsroutine Platz reserviert:



In der Routine wird dabei entschieden, ob die Task am eigentlichen Beginn gestartet werden soll oder ob sie nach einer Unterbrechung wieder zum Laufen kommt: In diesem Fall werden Register- und Befehlszählerstand restauriert.

- 7 Steht einer beauftragten Task nicht genügend Hauptspeicherplatz zur Verfügung, so wird untersucht, ob es verdrängbare Tasks gibt (siehe - 3 und - 4). Wenn ja, so werden

diese - je nach Bedarf und beginnend bei der niedrigsten Priorität - verdrängt. Als verdrängbar werden diejenigen Tasks bezeichnet, die unterbrochen im Hauptspeicher liegen oder die durch SUSPEND bzw. REQUEST am Weiterlauf gehindert werden. Nicht verdrängbar sind Tasks, die sich im Zustand "E/A-unterbrochen" befinden.

- 8 Eine Task-Aktivierung erfolgt über einen nichtparametrierten Auftrag (siehe 2.3.1).

- 9 Sehr wesentlich ist die Einführung eines Task - Kontroll - Blocks (TKB), in dem folgende Informationen festgehalten werden:

  - Taskzustand
  - Angabe, ob eine Task durch Verdrängung einer unterbrochenen Task zum Laufen kommt; wenn ja, wird der Listenindex einer Liste, in der die Programmnummern der verdrängten Tasks stehen, im TKB abgespeichert.

2.3.6 Beschreibung der gewählten Lösung

Die Lösung wird entsprechend der Vorgehensweise der Anpassung erläutert, nämlich

- Beschreibung der Betriebssystem-Module für die Realisierung der Tasking-Anweisungen
- Beschreibung der Modifikationen in der bisherigen Ablauforganisation aufgrund der Einführung obiger Module.

Stellvertretend für die Betriebssystemroutinen soll diejenige zur Realisierung der TERMINATE - Anweisung vorgestellt werden, einschließlich der dazu notwendigen zusätzlichen Listen und Unterprogramme. Durch die Gegenüberstellung mit dem bisherigen Betriebssystembaustein zur Beendigung einer Task soll der Aufwand und die Vorgehensweise der Erweiterung gezeigt werden.

Beendigung von Tasks mit der bisherigen Ablauforganisation

Wie bereits erwähnt, ist es für eine Task unter dem Betriebssystem MARVIS nur möglich, sich selbst zu deaktivieren. Der prinzipielle Ablauf ist in Bild 2.6 dargestellt.

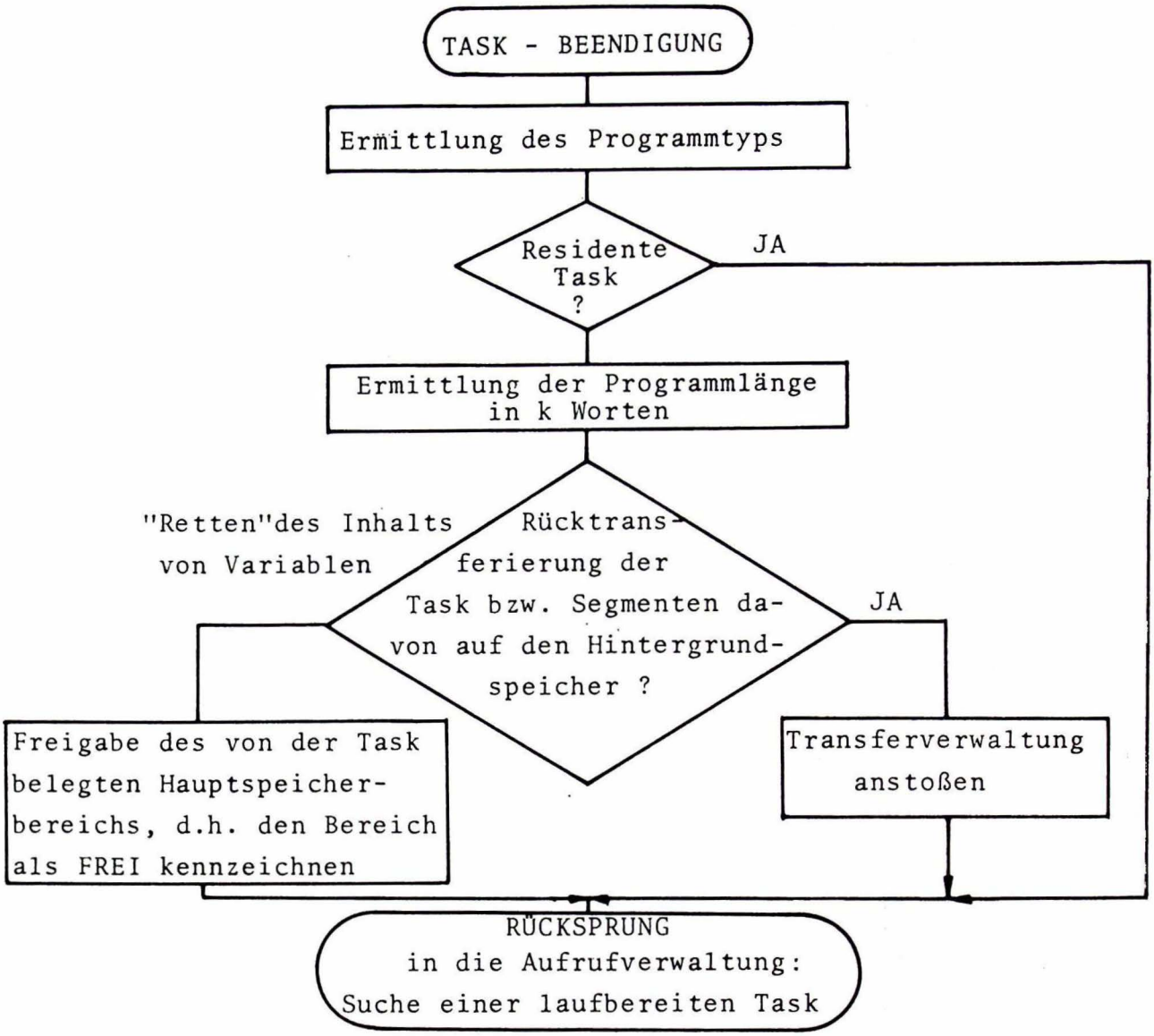


Bild 2.6: Ablaufdiagramm der MARVIS-Taskbeendigung

Beendigung von PEARL-Tasks

Im Gegensatz zum vorher beschriebenen Ablauf bietet PEARL die Möglichkeit, fremde Tasks zu deaktivieren, sofern diese nicht bereits im Zustand "ruhend" sind.
Zum Beenden einer Task, die sich im Zustand "E/A-unterbrochen" befindet, ist folgende Bemerkung zu machen:
Da es im Betriebssystem MARVIS nicht erlaubt ist, eine Task, die gerade auf eine Gerätefertigmeldung wartet, unmittelbar auf die TERMINATE-Anweisung hin zu deaktivieren, wird ein solcher Wunsch im TKB vorgemerkt (dasselbe gilt für SUSPEND). Nach Eintreffen der Geräterückmeldung nimmt dann das Betriebssystem die Task-Terminierung vor, einschließlich dem Löschen der Vormerkung.

Um den gesamten Betriebssystembaustein zur Realisierung der TERMINATE-Anweisung zu implementieren, mußten zusätzliche Listen und Unterprogramme eingeführt werden.

Die Liste PELUNT enthält die Programmnummern verdrängter Tasks (Bild 2

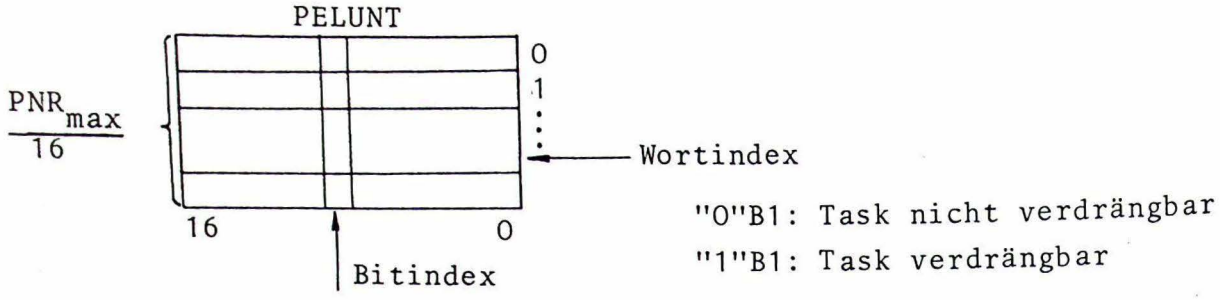


Bild 2.7: Aufbau der Liste PELUNT

Berechnung von Wort- und Bitindex aus der Programmnummer (PNR
Wortindex = ENTIER(PNR/16) ; Bitindex = Rest von PNR/16

Berechnung der PNR aus Wort- und Bitindex:
PNR = Wortindex·16 + Bitindex

Beim Beenden von Tasks des Zustandes "blockiert2" ist es erforderlich, diese aus den entsprechenden Warteschlangen zu nehmen. Auf die verschiedenen semaphorespezifischen Warteschlangen wird dabei mittels Zeiger verwiesen, die in der Liste SEMADR stehen (Bild 2.8):

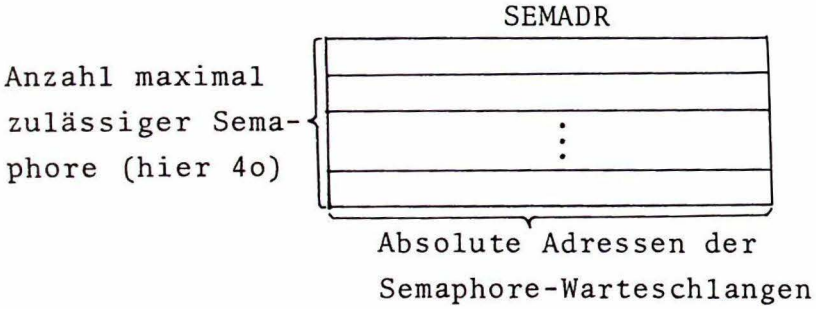


Bild 2.8: Liste SEMADR

Zur Ermittlung der Programmnummer bei vorliegender Tasknummer wurde die Liste PELTNR eingeführt (Bild 2.9):

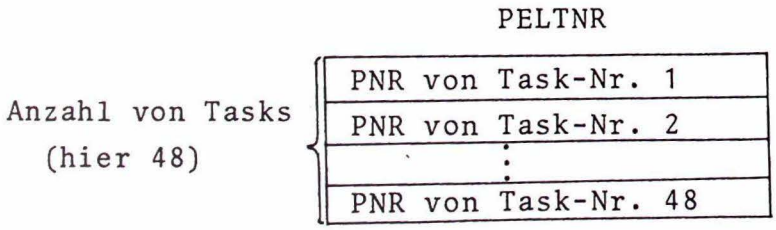


Bild 2.9: Liste PELTNR

Neben diesen Listen mußten drei Unterprogramme geschrieben werden, deren Aufgaben dem jeweiligen Flußdiagramm zu entnehmen sind (Bild 2.10).

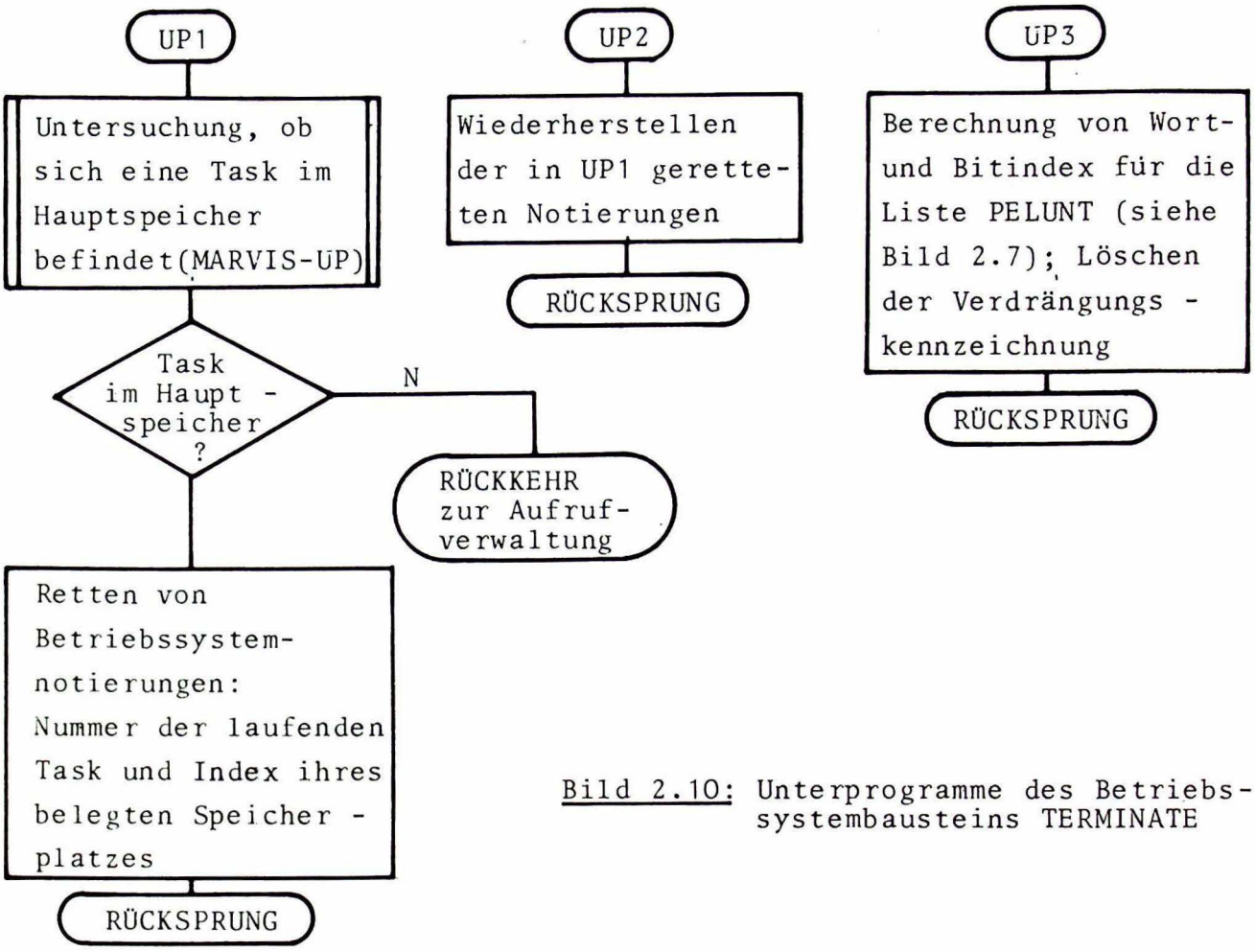


Bild 2.10: Unterprogramme des Betriebs- systembausteins TERMINATE

Unter Verwendung der zusätzlichen Listen und Unterprogramme ergibt sich der vollständige Modul TERMINATE (Bild 2.11). Man erkennt daraus deutlich den Verwaltungsaufwand, der nötig ist, fremde Tasks zu beenden. Hervorgerufen wird er im wesentlichen durch die verschiedenartigen Aufgaben bei unterschiedlichen Taskzuständen. Zahlenmäßig spiegelt sich die Erweiterung des Betriebssystembausteins TERMINATE in einer Speicherplatzzunahme von 130 % gegenüber dem Assemblerbaustein wieder.

Neben der ausführlichen Beschreibung dieses Moduls soll auf die wesentlichen Punkte der Eingriffe in die bisherige Ablauforganisation eingegangen werden. Wie bereits erwähnt, wurde dies aufgrund der Einführung der zusätzlichen Betriebssystemroutinen notwendig.

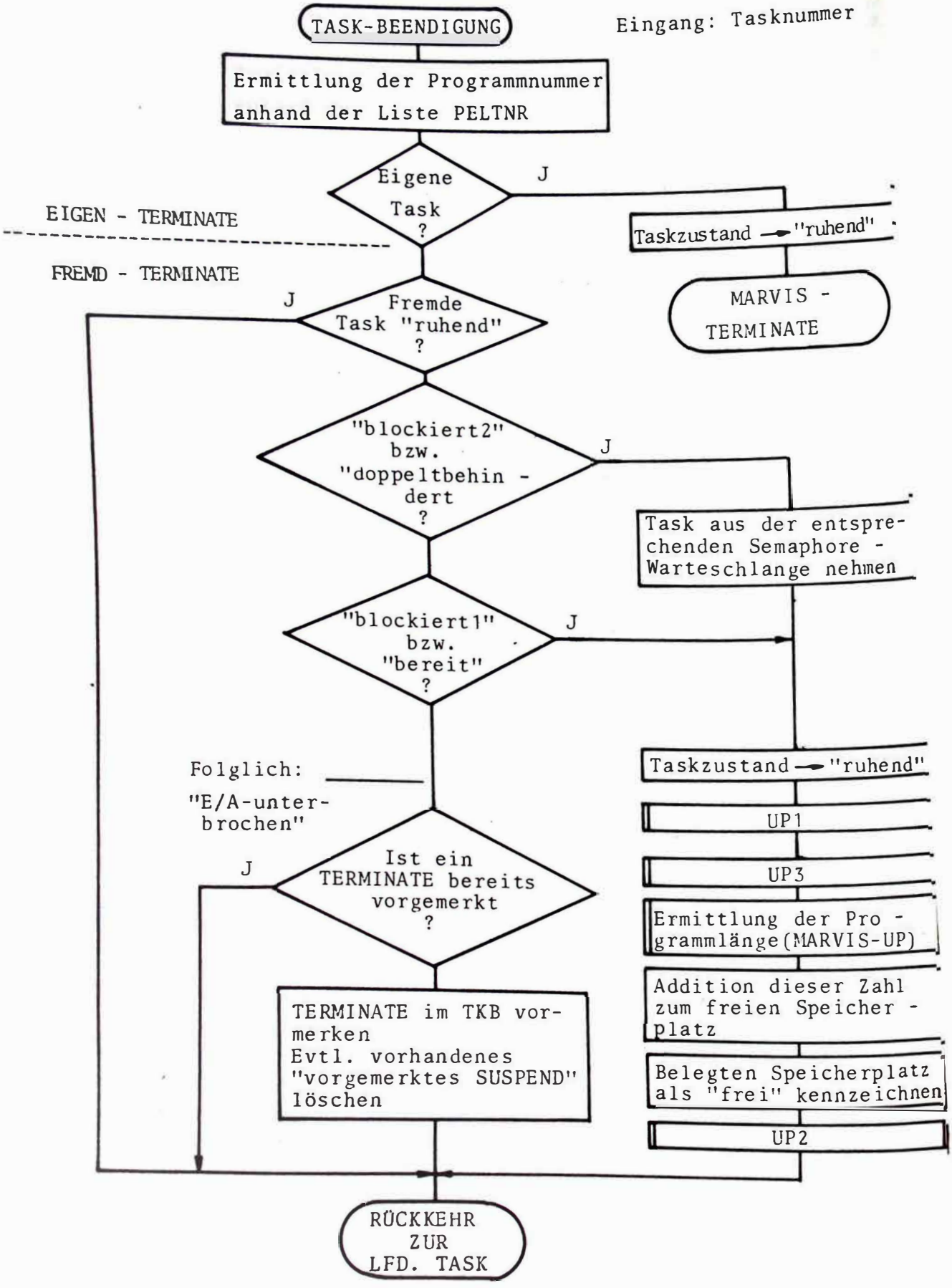


Bild 2.11: Betriebssystem-Modul TERMINATE

1. Änderung der Prozessorvergabe

Aus der Aufteilung der Klasse des Programmtyps B in mehrere Unterklassen folgen nachstehende neue Aufgaben:

- Feststellung, ob die beauftragte Task eine PEARL-Task ist (s. 2.3.5. - 1)
  Wenn ja: Untersuchung, ob ein Unterbrechen einer laufenden Task infolge einer laufbereiten, höherprioren Task erfolgen muß (die Taskprioritäten stehen in einer neu eingeführten, task-spezifischen Liste).
  - Wenn ja:-Taskzustand der angehaltenen Task in "bereit" ändern
    -Registerinhalte zum Taskkopf umspeichern
    -Task als verdrängbar in der Liste PELUNT markieren (siehe Bild 2.7)
  - Wenn nein: Fortsetzen der angehaltenen Task.

2. Verdrängung unterbrochener bzw. blockierter Tasks aus dem Hauptspeicher

Diese zusätzliche Aufgabe wurde durch das Zulassen mehrerer Prioritätsebenen innerhalb des Programmtyps B bzw. die Einführung der Taskzustände"blockiert1" bzw. "blockiert2" erforderlich.
Da MARVIS die Verdrängung von Tasks des Typs B nicht unterstützt, mußte ein vollkommen neuer Betriebssystem-Modul VERDRÄNGUNG (Bild 2.12) realisiert werden. Eine Verdrängung ist dann notwendig, wenn eine beauftragte PEARL - Task zu ihrem Lauf nicht genügend Hauptspeicherplatz verfügbar hat - Voraussetzung, daß es ausräumbare Tasks gibt. Diese werden, je nach Bedarf und beginnend mit der niedrigsten Priorität, auf den Hintergrundspeicher verdrängt und bei der Fortsetzung wieder in den Hauptspeicher transferiert, wobei der neue Laufbereich i.a. nicht mit dem ursprünglichen Laufbereich identisch ist. Daher ist vor dem Verdrängen die Fortsetzungsadresse zu relativieren.
Der Einbau des Moduls VERDRÄNGUNG erfolgt an derjenigen Stelle der Aufrufverwaltung, an der festgestellt wird, daß eine Task für ihren Lauf nicht genügend Speicherplatz zur Verfügung hat (Hauptspeicherverwaltung). Um ein zu häufiges Transferieren zu vermeiden, wurde der Baustein so verwirklicht, daß Tasks, die infolge der Beauftragung höherpriorer Tasks ausgeräumt wurden, erst dann fortgesetzt werden können, wenn für Sie wieder ein genügender Laufbereich zur Verfügung steht. Das bedeutet, daß sie zu ihrer Fortführung nicht ein weiteres Verdrängen erzwingen können: "Verdrängungstiefe" eins.

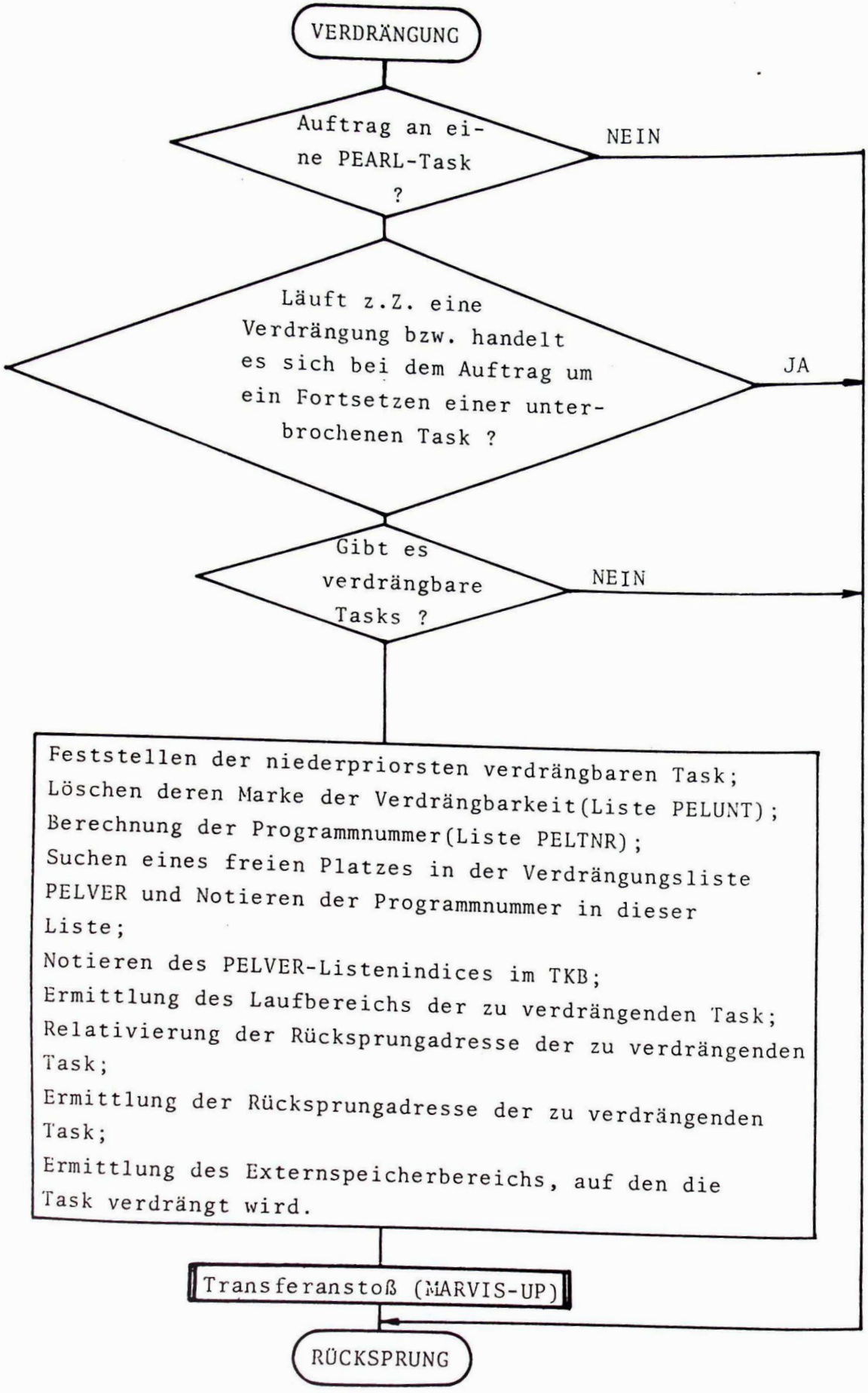


Bild 2.12: Betriebssystem-Modul VERDRÄNGUNG

Neben dem zusätzlichen Speicherplatz für das Ermöglichen von Verdrängungen, nämlich ca. 200 Maschinenworte, ist ein beträchtlicher Aufwand an Implementierungszeit zu rechnen.
Diese Zeit hängt sehr stark von der Kenntnis des Zielrechners ab:
Ein ausführliches Studium des Zielrechner-Betriebssystems ersparte insbesondere an dieser Stelle das mühsame Austesten mehrerer Versuchsversionen.

2.4 Ergebnisse der Adaption der Ablauforganisation

Aus der Beschreibung von MARVIS bzw. der Erweiterungen kann unmittelbar abgeleitet werden, welche wesentlichen Merkmale eine Ablauforganisation zur Erfüllung der PEARL - Anforderungen aufweisen sollte, um den Anpassungsaufwand des Zielrechner-Betriebssystems möglichst gering zu halten. Sie entsprechen weitgehend den sog. obligatorischen Elementarfunktionen von [18] , nämlich:

1. Funktionen zur Unterbrechungsbearbeitung
   - Retten/Restaurieren der Registerzustände(zum/vom Taskkopf bzw. einen "Keller")
   - Feststellen der Unterbrechungsursache (Auftraggeber).
2. Funktionen zur Taskverwaltung und Synchronisation
   - Module zur Realisierung der Tasking-Anweisungen, d.h. Verwaltung der in 2.3.2 angegebenen Taskzustandsänderungen sowie Führung des TKB; notwendige Module sind:
     - AKTIVIERUNG (ACTIVATE)
     - BLOCKIERUNG (SUSPEND)
     - FORTSETZUNG (CONTINUE)
     - BEENDIGUNG (TERMINATE)
   - Module zur Realisierung der Synchronisieranweisungen
     - Belegen einer Semaphore-Variablen (REQUEST)
     - Freigeben einer Semaphore-Variablen (RELEASE)

     Die damit zusammenhängenden Aufgaben wie das Führen des TKB durch die Zustandsänderungen von "laufend" nach "blockiert2", "blockiert2" nach "bereit", "doppeltbehindert" nach "blockiert1" sowie das Organisieren der semaphorespezifischen Warteschlangen der blockierten Tasks müssen verwaltet werden.

- Modul zur Lösung der Aufgabe "Auftraggeberidentifikation" [19] evtl. Modul zum Senden bzw. Entnehmen von Nachrichten zwischen Tasks
- Modul Prozessorvergabe:
  Suchen der höchstprioren laufbereiten Task, d.h. Belegen des Prozessors und Führung des TKB (Übergang "bereit" nach "laufend'

3. Funktionen zur Hauptspeicherverwaltung

- Modul zum Suchen und Belegen eines freien Hauptspeicherplatzes. Eingangsgröße dieses Moduls ist der benötigte Speicherplatz, Ausgangsgröße ist die Anfangsadresse des vorgeschlagenen Bereiches bzw. die Mitteilung, daß ein solcher Bereich z.Z. nicht verfügbar ist.
- Modul zum Suchen und Ausräumen verdrängbarer Tasks bzw. Segmenten davon
- Modul, der für das Nachladen verdrängter Tasks sorgt
- Modul zur Freigabe eines Hauptspeicherbereiches gewisser Länge und vorgegebener Anfangsadresse.

2.5 Zusammenfassung

In den letzten Abschnitten wurde die Vorgehensweise der Adaption der Ablauforganisation an PEARL am Beispiel der APS1-Implementierung auf der AEG 60-50 vorgestellt. Die dabei ausgeführten Teilaufgaben können wie folgt zusammengefaßt werden:

- Aufstellen der Anforderungen der betreffenden Programmiersprache an ein Betriebssystem
- Eingehende Untersuchung der verschiedenen Lösungsmöglichkeiten nach den erwähnten Kriterien
- Festlegung des Gesamtkonzepts (Übersicht). Klare Definition der Schnittstellen zwischen einzelnen Betriebssystemverwaltungen, z.B. zwischen der Ablauforganisation und der Ein/Ausgabeverwaltung.
- Beschreibung des Ablaufs neuer Betriebssystem-Moduln und der damit verbundenen Änderungen im bisherigen Betriebssystem
- Implementierung der Betriebssystem-Erweiterungen einschließlich Test.

3. Aufbau des Modellprozesses "Betriebsüberwachung eines Fließprozesses"

3.1 Aufgaben eines Modellprozesses bei der Sprachuntersuchung

In der Einführung wurde die Bedeutung eines Modellprozesses bei der Sprachuntersuchung herausgestellt.
Die Aufgaben eines Modells lassen sich in zwei Teile gliedern [21]:

a) Gegenstand zur Ermittlung typischer Automatisierungsaufgaben der betreffenden Aufgabenstellung - hier Betriebsüberwachung eines Fließprozesses. Das gegenständliche Modell bildet also die Grundlage, aus der sich die Sprachanforderungen ergeben.

b) Gegenstand zur Überprüfung der Programme, die in der zu untersuchenden Sprache formuliert werden:
Vergleich des Programmablaufs mit dem Prozeßablauf, d.h. Untersuchung ihres Zusammenwirkens auf Richtigkeit. Anhand des Modells kann nachgewiesen werden, daß die Sprachanweisungen ordnungsgemäß ablaufen.

Aus diesen Aufgaben folgen die Anforderungen an ein Modell:

1. Generelle Anforderungen an einen Modellprozeß

   Zur Rechtfertigung der Bezeichnung "Modellprozeß" hat dieser allgemeinen Bedingungen zu genügen. Daneben werden an die Auswahl und den Aufbau eines gegenständlichen Modells gewisse Anforderungen gestellt, um eine wirksame Sprachuntersuchung zu gewährleisten.

2. Spezielle Anforderungen an den Modellprozeß "Betriebsüberwachung eines Fließprozessses"

   Der Modellprozeß muß die Merkmale des Prozeßtyps aufweisen - hier diejenigen eines Fließprozesses. Ferner muß gewährleistet sein, daß die charakteristischen Teilaufgaben der Aufgabenstellung - hier Betriebsüberwachung - anwendbar, d.h. demonstrierbar sind.

Nach einer Vorstellung dieser Anforderungen wird eine Beschreibung des Modellprozesses gegeben:

- Nachweis, daß das Modell den Anforderungen genügt
- Bezugsquelle der Sprachuntersuchung:
  Diese wird anhand von Beispielen für typische Überwachungsaufgaben am Modell durchgeführt und setzt demgemäß die Kenntnis des Prozesses voraus.

3.2 Kriterien zur Ermittelung geeigneter gegenständlicher Modelle technischer Prozesse für eine Sprachuntersuchung

3.2.1 Generelle Anforderungen

Vor der Auswahl und dem Aufbau des Modellprozesses ist zu untersuchen, welche generellen Anforderungen an einen Modellprozeß zu stellen sind:

- Echter, wirklichkeitsnaher Prozeß mit den charakteristischen Eigenschaften des Prozeß-Grundtyps. Die Prozeßmeß- und Stellglieder sollen den in industriellem Einsatz befindlichen entsprechen bzw. dieselben sein.
- Überschaubarer, einfacher Aufbau, damit die Funktionen des Prozesses im Zusammenwirken mit dem Programmsystem überblickt werden können.
- Flexibilität
  Für abgeänderte bzw. neue Aufgabenstellungen soll die Anlage leicht modifizierbar sein.
- Sicherheit
  Bei einer falschen Bedienung der Anlage bzw. einer Störung dürfen keine gefährlichen Prozeßzustände auftreten.

- Störungsunempfindlichkeit
  Die Anlage soll unempfindlich gegenüber elektrischen und mechanischen Störungen sein. Tritt ein Schaden auf, soll der defekte Teil leicht reparierbar bzw. austauschbar sein.
- Prozeßkenntnis
  Der Wertebereich der Prozeßvariablen bzw. ihr zeitlicher Verlauf sollte möglichst von vornherein bekannt sein.

Die wichtigsten Randbedingungen bei der Auswahl eines Modellprozesses sind folgende:

- Kosten
  - für den technischen Prozeß, die notwendigen Meß- und Stellglieder, Verstärker, Netzgeräte usw.
  - Personalkosten für den mechanischen und elektrischen Aufbau.
- Zeit für den Modellprozeß-Aufbau
  Dieser Punkt steht in engem Zusammenhang mit den vorhandenen finanziellen Mitteln. Je größer diese sind, umso mehr kann auf die Selbstherstellung von Baueinheiten verzichtet werden, was den Zeitaufwand entsprechend verringert.
- Umgebungsbedingungen
  Unter diesem Gesichtspunkt sind insbesondere die zur Verfügung stehenden Platzverhältnisse in einem Laboratorium zu verstehen, ebenso wie eventuell eine Wasseranschlußmöglichkeit, Druckluft- bzw. Spannungsversorgung usw .
- Qualifikation des Personals
  Ein Modellprozeß wird umso einfacher sein müssen, je weniger qualifiziert das zum Aufbau und zur Bedienung sowie evtl. zur Wartung vorhandene Personal ist.
- Transportfähigkeit
  Beim Aufbau der Anlage ist darauf zu achten, diese für Vorführungen und Untersuchungen an anderen Orten transportfähig zu halten.

Aus den hier genannten Kriterien und Randbedingungen ergibt sich die Alternative

1. Selbstbau des Modellprozesses
2. Kauf des Modellprozesses

Auf die erstgenannte Möglichkeit muß dann zurückgegriffen werden, wenn kein geeignetes Modell käuflich erhältlich ist[+)] bzw. die finanziellen Mittel dazu fehlen. Der wesentliche Nachteil dieser Alternative liegt im Zeitaufwand und - bei nicht sehr sorgfältigem Aufbau - in einer evtl. Störungsempfindlichkeit bzw. fehlender Sicherheit. Der Vorteil ist darin zu sehen, daß durch die Verwendung des Modellprozesses nur für den speziellen Zweck der Sprachuntersuchung dieser entsprechend flexibel aufgebaut werden kann und nur mit den notwendigen Einrichtungen versehen werden muß.

Zur zweiten Lösungsmöglichkeit, also dem Kauf eines Modells, ist folgendes zu bemerken:
Der Nachteil liegt in einer evtl. Unübersichtlichkeit

- aufgrund von Anlagenteilen, die für die Sprachuntersuchung nicht benötigt werden
- durch den Einbau zusätzlicher Meß- und Stellglieder bei beengten Platzverhältnissen.

Demgegenüber steht eine Reihe von Vorteilen:

- Schnelle Realisierung des Aufbaus. Den Einbau von Meß- und Stellgliedern kann i.a. der Hersteller übernehmen, was auch bei dem anschließend beschriebenen Modell geschehen ist. Durch seine Anlagenkenntnis kann er die notwendigen Angaben über erforderliche zusätzliche Bauteile machen und deren fachgerechte Montage vornehmen.

  Beispiel Modellprozeß "Betriebsüberwachung":
  Nach der Ermittlung der geeigneten Fühler im Rahmen einer Studienarbeit [*20*] erfolgte der Einbau in ca. 3 Tagen.

- Prozeßkenntnis
  Die genaue Funktionsweise der Anlage ist dem Hersteller bekannt und kann in Form von verbalen Beschreibungen oder Diagrammen übernommen werden.

- Eine käufliche Anlage wird normalerweise den Sicherheitsanforderungen genügen[++)], auch eine Störungsunempfindlichkeit sowie die Wartung der Anlage ist i.a. gewährleistet.

+) Für eine gegenständliche Nachbildung eines Stückprozesses war kein Modell käuflich erhältlich. Daher wurde ein Modellprozeß "Materialtransportsystem mit Hochregallager" aufgebaut.

++) hier TÜV-Prüfung

3.2.2 Spezielle Anforderungen an den Modellprozeß "Betriebsüberwachung eines Fließprozesses"

Zeitverhalten der Prozeßgrößen

Voraussetzung für den Modellprozeß bezüglich des Prozeßgrundtyps ist ein Modell mit den Kennzeichen von Fließprozessen, also zeit- bzw. zeit- und ortsabhängigen Prozeßvariablen mit kontinuierlichem bzw. stückweise kontinuierlichem Wertebereich.

Anforderungen an den Modellprozeß durch die Aufgabenstellung Betriebsüberwachung

Die gewählte, typische Aufgabenstellung aus dem Bereich der Automatisierung mit Prozeßrechnern erfordert ein gegenständliches Modell, an dem die charakteristischen Eigenschaften der Betriebsüberwachung demonstrierbar sind[*22*] ,[*23*] ,[*24*] und[*25*] .

Gemäß [*22*] läßt sich definieren:

"Prozeßüberwachung heißt, den Ablauf des technischen Prozesses ständig zu kontrollieren, Abweichungen vom ordnungsgemäßen Ablauf festzustellen und ggf. Maßnahmen einzuleiten, um zu verhindern, daß Betriebszustände eintreten können, die Leib und Leben von Menschen, den Bestand und die Funktionsfähigkeit der technischen Anlagen oder die Qualität des Produkts gefährden."

Anforderungen an den Prozeßablauf

Die Automatisierungsprogramme haben die Aufgabe, irreguläre Betriebszustände festzustellen und zu dokumentieren.
Anhand der Dokumentation können die Überwachungsprogramme auf ihre Richtigkeit hin überprüft werden, d.h. es kann festgestellt werden, ob bei einer entsprechenden Störung die zugeordnete Programmreaktion folgt.

Irreguläre Prozeßabläufe lassen sich gemäß den verschiedenen Ursachen aufteilen in:

- Irreguläre Prozeßabläufe aufgrund von
  - Prozeßstörungen
  - Ausfällen
  - Fehlbedienung
- Scheinbar irreguläre Prozeßabläufe aufgrund von
  - Störungen bzw. Ausfall von Meßeinrichtungen (z.B. Spannungsausfall)
  - Störung der Signalübertragung.

Aus Gründen der Sicherheit - eine Rechnerstörung soll nicht zu einem gefährlichen Prozeßzustand führen - wird die überwiegende Zahl von Prozeßrechnern im offen prozeßgekoppelten Betrieb (on line open loop) eingesetzt. Die Eingriffsverantwortung bleibt dabei beim Bedienungspersonal, d.h. es werden vom Rechner keine Stellgrößen ausgegeben. Diese typische Betriebsart wurde auch bei der Überwachung des Modellprozesses angewandt, wobei sich dann Bild 3.1 ergibt.

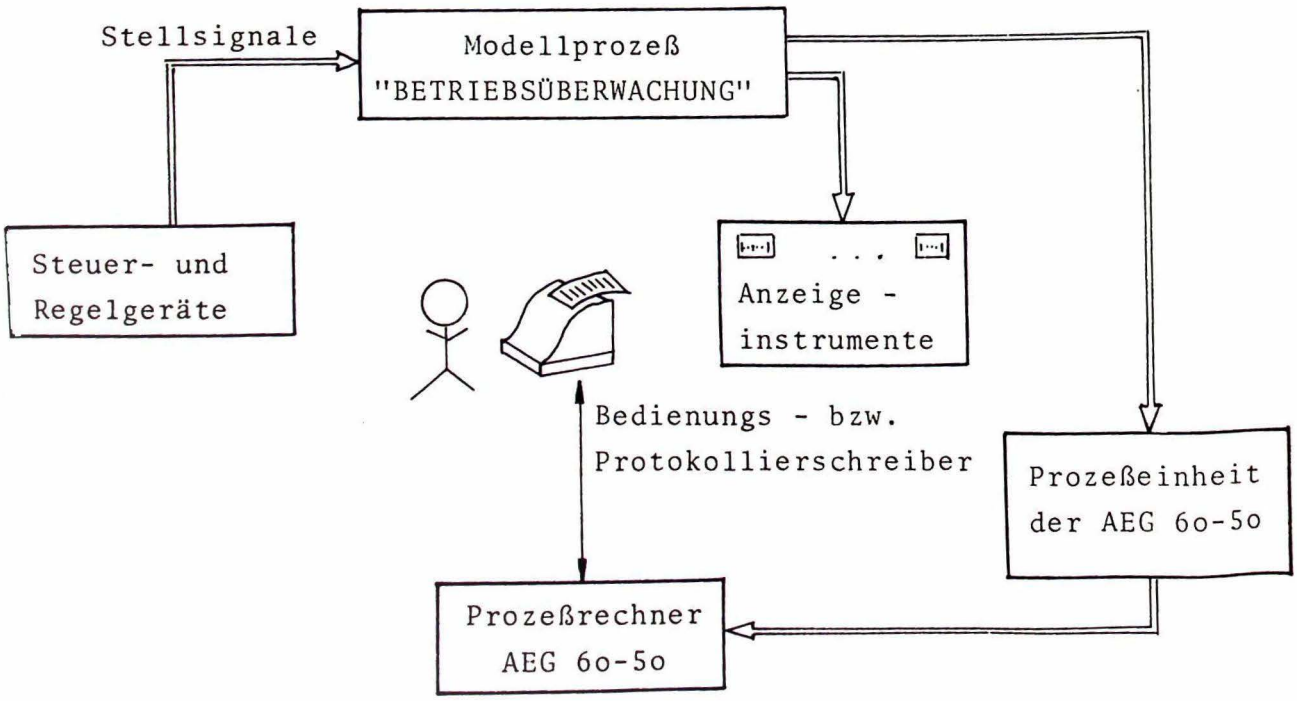


Bild 3.1: Offen prozeßgekoppelter Betrieb Modellprozeß - AEG 60-50

3.3 Aufbau und Wirkungsweise des Modellprozesses

Als Modell wurde ein Kaffeeautomat verwendet, der zu diesem Zweck umgebaut und mit den entsprechenden Meßfühlern versehen wurde[+].

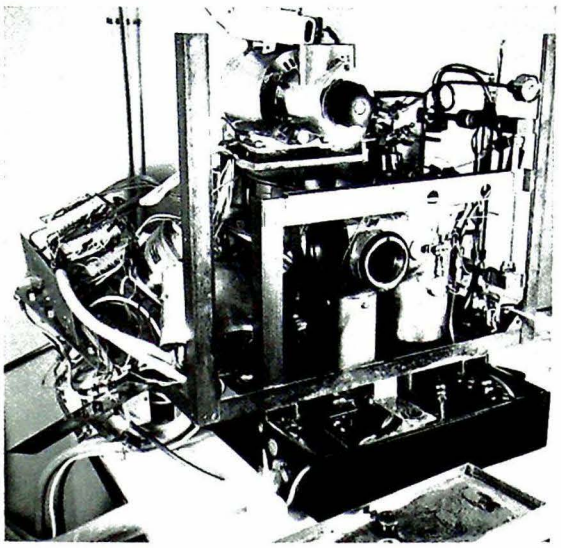


Bild 3.2: Umbau des Modellprozesses und Einbau der Fühler

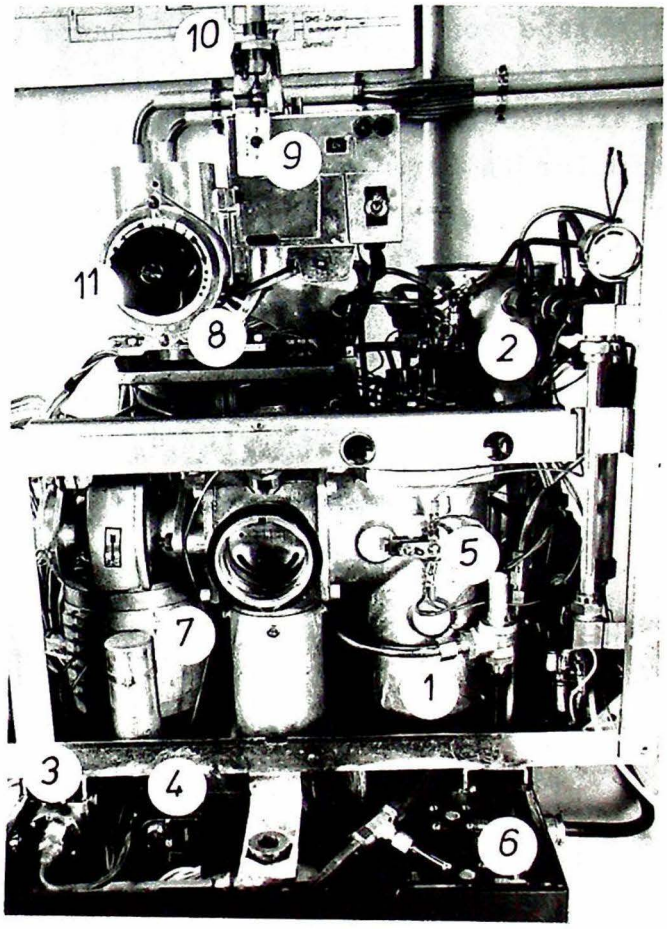


Bild 3.3: Vorderansicht des Modellprozesses bei abgenommener Frontplatte (Die Zahlen werden im Text erklärt).

+) Vom Hersteller des Kaffeeautomaten werden bei der Entwicklung ähnliche Meßfühler eingebaut. Es handelt sich also bei dem hier gewählten Modellprozeß gleichzeitig um einen echten Anwendungsfall aus der Praxis.

Der Prozeß beinhaltet einen Wasserkessel (1) sowie einen Dampfkessel zur Abgabe von Heißwasser bzw. Dampf (2). Zur Überwachung der Betriebsgrößen Druck und Temperatur verfügen beide Kessel über Druckfühler (Dehnungsmeßstreifen bzw. induktiver Fühler) sowie Temperaturfühler in Form von Thermoelementen und Widerstandsthermometern. Zur Erwärmung des Wassers besitzen beide Kessel Heizspiralen, die vom Netz gespeist und über Druck(3)- bzw. Temperaturregler (4) eingeschaltet werden. Hierbei werden die Heizströme überwacht. Ein Trockengschutz (5) verhindert, daß bei leerem Kessel die Heizung eingeschaltet wird. Der Wasserzulauf in die Kessel erfolgt über schwimmergesteuerte Magnetventile (6) und eine Pumpe (7). Prozeßgrößen, die in diesem Zusammenhang erfaßt werden, sind

- Dampfkessel-Füllstand (kapazitiver Geber)
- Wasserleitungsdruck (Dehnungsmeßstreifen-Geber)
- Durchflußmenge (Flügelradgeber)
- Pumpenmotorstrom.

Das Mahlgut wird mit Hilfe eines Drehstrommotors gemahlen (8), dessen aufgenommener Strom und Einschaltzeitdauer überwacht werden. Diese Laufzeit wird durch das Zählen von Impulsen konstanter (Netz-) Frequenz ermittelt; die Zeit hängt dabei von der Stellung eines verschiebbaren Endschalters ab (9), die zu diesem Zweck mit einem induktiven Weggeber (10) erfaßt wird. Der Mahlgrad (Körnigkeit) kann mit einem Drehknopf (11) beeinflußt werden. Ein induktiver Drehwinkelgeber mißt die eingestellte Position.
Das Wirkschaltbild des Modells wird in Bild 3. 4 gezeigt. Anlagenteile, die aus Gründen der Übersichtlichkeit dort nicht eingezeichnet sind, werden in der noch folgenden Beschreibung der Prozeßgrößen aufgeführt.

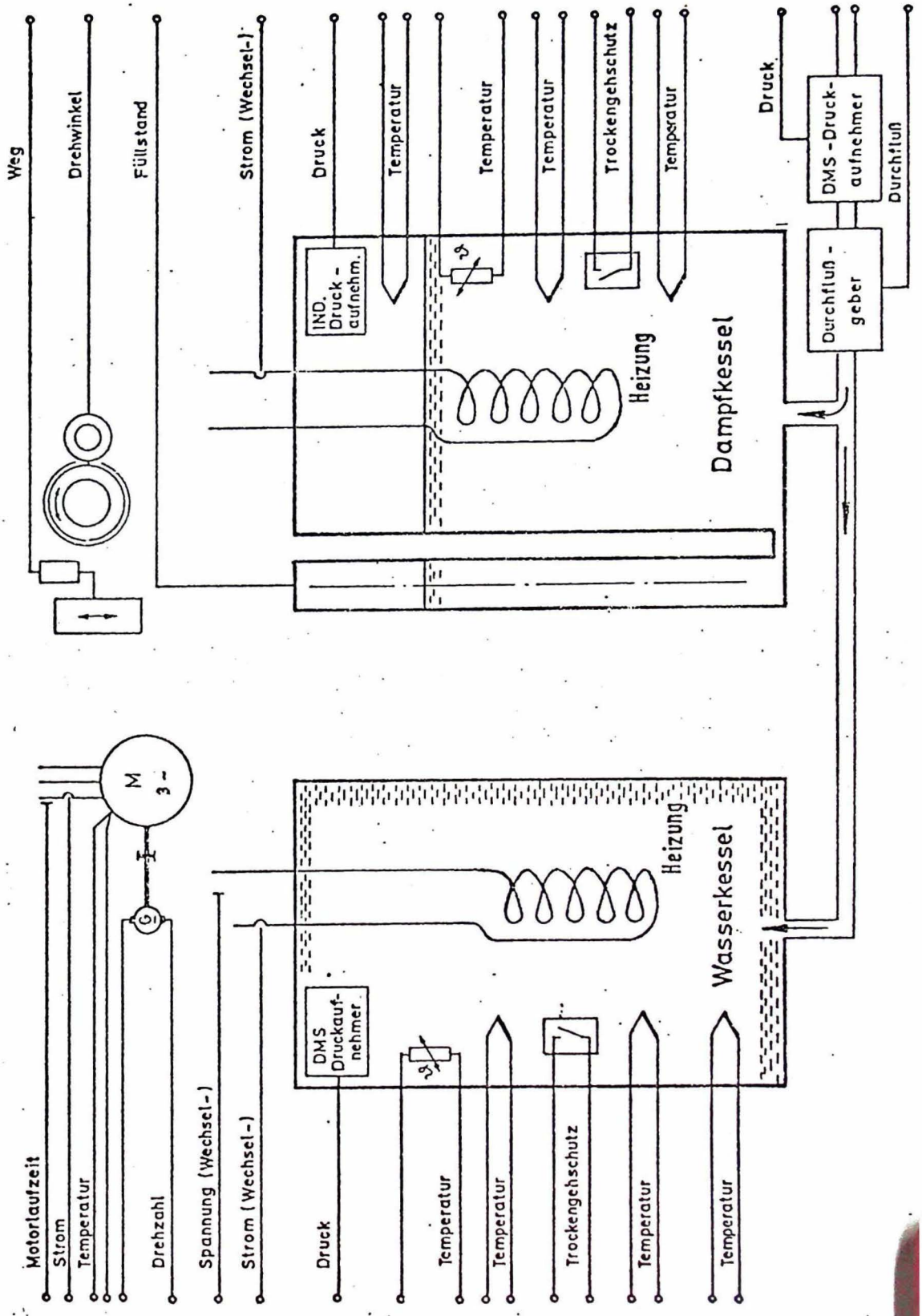


Bild 3.4: Modellprozeß "Betriebsüberwachung"

Zum Vergleich mit den vom Rechner erfaßten Werten werden diejenigen der wichtigsten Analoggrößen auf Anzeigeinstrumenten dargestellt (Bild 3. 5) , während der Zustand der Binärgrößen mittels Glimmlampen angezeigt wird (Bild 3. 6) . Daneben dient das Anzeigenfeld der manuellen Überwachung der Anlage bei einem Rechnerausfall.

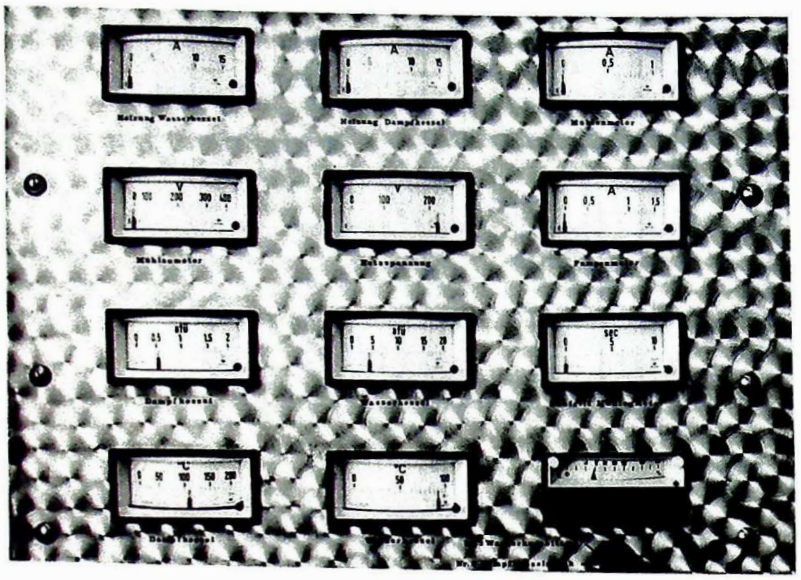


Bild 3.5: Anzeigeinstrumente für die wichtigsten Analoggrößen

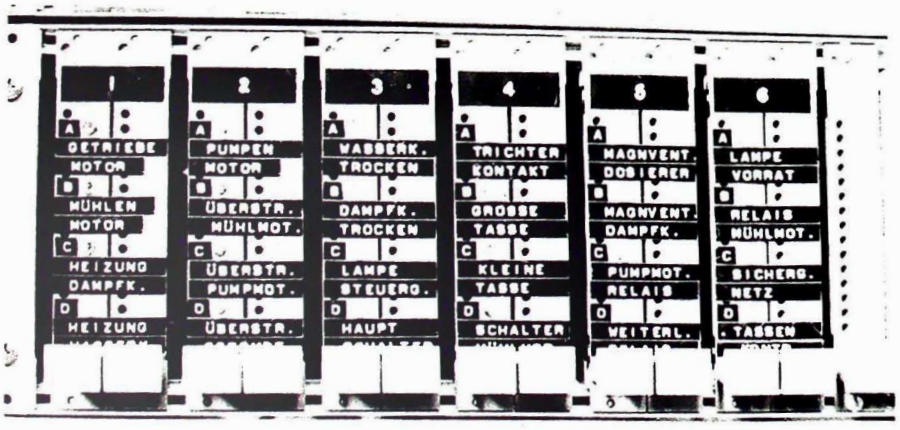
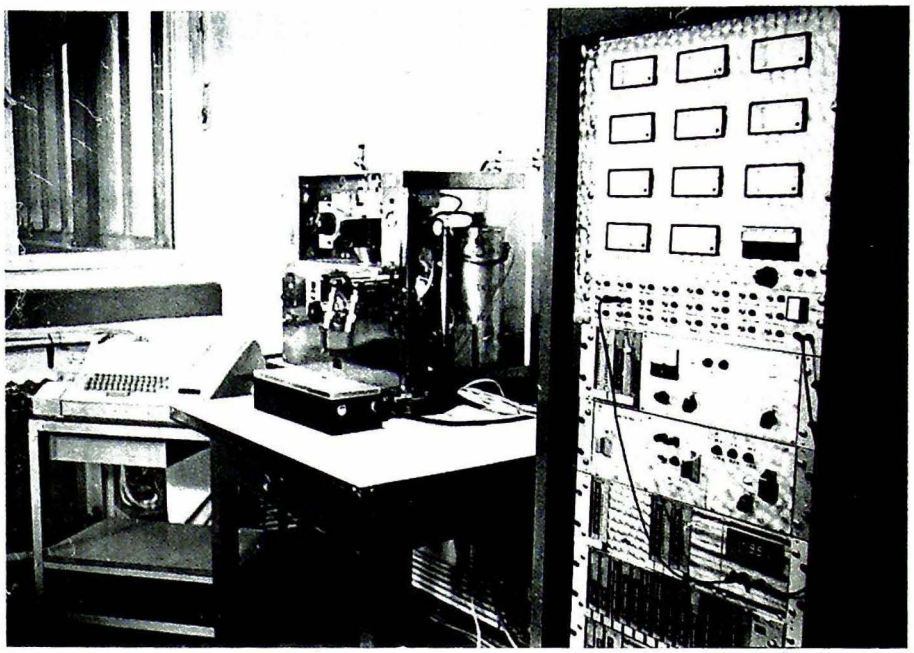


Bild 3.6: Anzeige der Binärzustände

Der Gesamtaufbau des Modellprozesses ist in Bild 3.7 zu sehen:



Modellprozeß "Betriebsüberwachung"

Bedienungsschreibmaschine

Verstärkerschrank mit
- Anzeigeinstrumenten für Analoggrößen
- Glimmlampen zur Binärgrößenanzeige
- Vorverstärker mit Netzgeräten

Bild 3.7: Gesamtaufbau des Modellprozesses

3.4 Übersicht über die zu überwachenden Größen des Modellprozesses

Im letzten Abschnitt wurden die wesentlichen Teile sowie die Funktionsweise des Modells geschildert. Für eine Gegenüberstellung der Anforderungen an ein solches Modell eines Fließprozesses und der gewählten Lösung ist es notwendig, die zu überwachenden Größen, ihre Signaldarstellung und den Anschluß an die Prozeßeinheit zu kennen. Außer diesem Gesichtspunkt erfordert die Sprachuntersuchung, d.h. die Anwendung der typischen Überwachungsaufgaben auf den Modellprozeß die Kenntnis dieser Informationen.
Die dazu notwendigen Angaben werden - getrennt nach Analoggrößen und Binärgrößen - auf den folgenden Seiten zusammengestellt.

Prozeßgröße		Beschreibung der Größe	Überwachung	Signal-darstellung
Temperatur [°C]	Wassertemperatur Dampfkessel	Über eine Zweipunkt-Druckregelung wird die Wassertemperatur - je nach Wasser- bzw. Dampfentnahme - mehr oder weniger konstant gehalten. Zeitpunkt und Menge des entnommenen Wassers bzw. Dampfs sowie der Zeitpunkt des Ein- und Ausschaltens der Heizung bestimmen den Temperaturverlauf.	zyklisch	amplituden-analog
	Dampftemperatur Dampfkessel	s.o. (anstelle Wassertemperatur: Dampftemperatur)	zyklisch	amplituden-analog
	Wassertemperatur Wasserkessel	s.o.	zyklisch	amplituden-analog
Druck [atü]	Wasserkesseldruck	Der Wasserkesseldruck wird in erster Linie vom Ein- bzw. Ausschalten der Heizung sowie der entnommenen Wassermenge bestimmt.	zyklisch	amplituden-analog
	Dampfkesseldruck	Über die Zweipunktregelung wird der Kesseldruck - je nach Wasser- bzw. Dampfentnahme - mehr oder weniger konstant gehalten. Zeitpunkt und Menge des entnommenen Wassers bzw. Dampfs sowie der Zeitpunkt des Ein- und Ausschaltens der Heizung bestimmen den Druckverlauf.	zyklisch	amplituden-analog
	Wasserzuleitungs-druck	Konstanter Wert im stationären Betrieb; Störungen durch Druckschwankungen in der Leitung bzw. durch das Schalten der Magnetventile für den Wasserzulauf in die Kessel beeinflussen den Druckverlauf.	zyklisch Berücksichtigung des Wasserzulaufs in die Kessel	amplituden-analog

Prozeßgröße	Beschreibung der Größe	Überwachung	Signal-darstellung
Durchfluß [dm^3/min]	Der Durchfluß wird bestimmt von der Wasserentnahmemenge im Wasser- bzw. Dampfkessel: Öffnen und Schließen zweier Magnetventile	zyklisch (bei Einschalten der Magnetventile	frequenz-analog Erfassung einer Impulsfrequenz (Digitaleingabe + Unterbrechungseingabe)
Dampfkessel-Füllstand [cm]	Über einen Schwimmer wird der Wasserzufluß in den Dampfkessel geregelt. Ein Füllstandsmesser überwacht den aktuellen Wasserstand.	zyklisch	amplituden-analog
Mühlen-Laufzeit [sec]	Die Zeit zwischen dem Einschalten des Motors durch Drücken eines EIN-Tasters und dem selbständigen Abschalten des Motors durch einen Endschalter wird mit einem Zeitmesser erfaßt.	azyklisch Erfassung nach Abschalten des Motors	frequenz-analog Erfassung eines Zählwertes (Digitaleingabe)
Mühlen-Drehzahl [1/min]	Die Motordrehzahl, die nach einer kurzen Anlaufzeit ihren konstanten Nennwert erreicht, wird während der Motorlaufzeit erfaßt.	zyklisch (bei Motor EIN; Beachtung der Anlaufzeit)	amplituden-analog
Wechselspannung [V]	Spannungsversorgung der Anlage	zyklisch	amplituden-analog

Prozeßgröße		Beschreibung der Größe	Überwachung	Signal-darstellung
Wechsel-strom [A]	Heizung Wasserkessel	Über einen Zweipunkt-Temperaturregler wird die Heizung ein- und ausgeschaltet.	zyklisch (bei Heizung EIN)	amplituden-analog
	Heizung Dampfkessel	Über einen Zweipunkt-Druckregler wird die Heizung ein- und ausgeschaltet.	zyklisch (bei Heizung EIN)	amplituden-analog
	Pumpenmotorstrom	Nach der Wasserentnahme aus dem Wasserkessel wird automatisch der Pumpenmotor für eine bestimmte Zeit eingeschaltet, um frisches Wasser in den Kessel zu leiten.	zyklisch (bei Motor EIN)	amplituden-analog
Mühlenmotor-Drehstrom [A]		Dieser Motor wird durch das Drücken eines EIN-Tasters für zwei bis vier Sekunden eingeschaltet.	zyklisch (bei Motor EIN)	amplituden-analog
Endschalterposition [cm]		Das Ende des Laufs der Mühle wird von der Position eines Endschalters bestimmt; aus diesem Grund wird die Position mit einem Weggeber erfaßt.	zyklisch	amplituden analog
Drehwinkel [Grd]		Erfassung einer Drehknopfstellung (Mahlgrad)	zyklisch	amplituden-analog

Prozeßgröße		Beschreibung der Größe	Überwachung	Signal-darstellung
EIN - Taster		Bei Drücken des EIN-Tasters wird die Wasserentnahme aus dem Wasserkessel eingeleitet sowie der Drehstrom-motor eingeschaltet.	azyklisch	binär (Unterbrech-ungseingabe)
Sperrkontakt		Über diesen Kontakt wird der EIN-Taster "verriegelt".	zyklisch	binär
AUS - Schalter		Das automatische Einschalten des Drehstrommotors über den EIN-Taster kann durch einen AUS-Schalter verhindert werden.	zyklisch	binär
Thermostat - Wasserkessel		Ein Zweipunkt-Temperaturregler sorgt für eine konstante Wasseraustrittstemperatur aus dem Wasserkessel: EIN- bzw. AUSschalten der Heizung	azyklisch	binär (Unterbrech-ungseingabe)
Druckregler - Dampfkessel		Ein Zweipunkt-Druckregler sorgt für konstanten Druck im Dampfkessel: EIN- bzw. AUSschalten der Heizung	azyklisch	binär (Unterbrech-ungseingabe)
Trocken-gehschutz	Wasser-kessel	Ein Trockengehschutz verhindert, daß bei zu niedrigem Wasserstand die Heizung eingeschaltet wird.	zyklisch	binär
	Dampf-kessel	s.o.	zyklisch	binär
Lichtschranke	Glasbehälter	Füllstandsüberwachung des Kaffee-Vorratsbehälters	zyklisch	binär
	Kaffee-Ent-nahmestelle	Überwachung der Kaffee-Entnahmestelle	azyklisch	binär

Prozeßgröße		Beschreibung der Größe	Überwachung	Signal-darstellung
Wasserzulauf-ventil	Wasser-kessel	Einleiten von frischem Wasser in den Wasserkessel nach der Wasserentnahme, d.h. bei bzw. nach dem Lauf des Pumpenmotors	zyklisch	binär (Unterbrechungseingabe)
	Dampf-kessel	Einleiten von frischem Wasser in den Dampfkessel, d.h. bei Anziehen des Niveaurelais. Das Ende des Wasserzulaufs wird mit dem Abfallen des Niveaurelais erreicht.	zyklisch	binär (Unterbrechungseingabe)
Überstrom-schutz	Drehstrom-motor	Bei Überschreiten des Stromes über einen gewissen Wert wird der Motor automatisch abgeschaltet.	azyklisch (nach Motorlauf)	binär
	Pumpen-motor	s.o.	azyklisch (nach Motorlauf)	binär
Hauptschalter		Ein- bzw. Ausschalten der Anlage	zyklisch	binär
Schlüsselschalter		Zusätzliche Verriegelung der EIN-Taste	zyklisch	binär
Netzsicherung		Anlagenabsicherung	zyklisch	binär

3.5 Reguläre Betriebsarten des Modellprozesses

Aus der Beschreibung des gegenständlichen Modells und seiner Variablen lassen sich 4 grundsätzlich verschiedene reguläre Betriebsarten definieren, auf die bei der Anwendung der Überwachungsaufgaben häufig Bezug genommen wird (Bild 3.8):

Betriebsarten		Merkmale
Anlaufphase	Wasserkessel	Einschalten der Wasserkesselheizung bis Solltemperatur erreicht; Wasserzufluß bis Kessel gefüllt
	Dampfkessel	Einschalten der Heizung bis Solldruck erreicht; Wasserzulauf in den Kessel bis Normalhöhe erreicht
Arbeitsphase	Keine Entnahme	Keine Entnahme von Kaffee, Heißwasser bzw. Dampf, kein Wasserzufluß. Infolge Wärmeabstrahlung der beiden Kessel sind die Temperaturen der Kessel zeit- u. ortsabhängig.
	Kaffeeentnahme +)	Drücken des EIN-Tasters, Einschalten des Mühlenmotors, Heißwasserentnahme, Kaltwasserzulauf in den Kessel über die Pumpe, Einschalten der Heizung
	Wasserentnahme (Dampfkessel) +)	Heißwasserentnahme über ein Ventil; Wasserzulauf in den Kessel, evtl. Einschalten der Heizung
	Dampfentnahme (Dampfkessel) +)	Dampfentnahme über ein Ventil, evtl. Wasserzulauf in den Kessel sowie Einschalten der Heizung

Bild 3.8: Betriebsarten des Modellprozesses

Unter Zuhilfenahme dieses Bildes und der Beschreibung der Prozeßvariablen läßt sich eine Zulässigkeitstabelle (Bild 3.9) aufstellen, die angibt, wann eine Erfassung überhaupt sinnvoll d.h. notwendig wird.

+) Diese Betriebsarten können zeitlich parallel auftreten!

Betriebsart / Prozeßgröße	Anfahrphase	Arbeitsphase		
		"Keine Entnahme"	"Kaffee-Entnahme"	"Wasser(Dampf-)entnahme Dampfkessel"
Wechselspannung	x	x	x	x
Wechselstrom WK +)	x	x	x	-
Wechselstrom DK +)	x	x	-	x
Wechselstrom Pumpe	-	-	x	-
Drehstrom Mühle	-	-	-	-
Temperatur WK	x	x	x	-
Temperatur Wasserzulauf	x	x	x	x
Wassertemperatur DK	x	x	-	x
Dampftemperatur DK	x	x	-	x
Druck WK	x	x	x	-
Druck DK	x	x	-	x
Druck Wasserzuleitung	x	x	x	x
Durchfluß	x	-	x	x
Mühlenlaufzeit	-	-	x	-
Mühlendrehzahl	-	-	x	-
Endschalterposition	-	-	x	-
Drehwinkel	-	-	x	-
Hauptschalter	x	x	x	x
Netzsicherung	x	x	x	x
Schlüsselschalter	-	-	x	-
Überstromschutz Pumpe	-	-	x	-
Überstromschutz Mühle	-	-	x	-
Wasserzulaufventil WK	x	x	x	-
Wasserzulaufventil DK	x	x	-	x
Trockengehschutz WK	x	-	x	-
Trockengehschutz DK	x	-	-	x
Thermostat WK	x	x	x	-
"Druckregler" DK	x	x	-	x
Lichtschranke	-	-	x	-
EIN - Taste	-	-	x	-
Sperrkontakt	-	-	x	-
AUS-Schalter	-	-	x	-

Bild 3.9: Zulässigkeitstabelle (+) WK = Wasserkessel, DK = Dampfkessel)

3.6 Irreguläre Betriebsabläufe

Wie bereits erwähnt, ist es die Aufgabe der Automatisierungsprogramme, den Prozeß auf reguläre Zustände hin zu überwachen. In Bild 3.1o werden die wesentlichen Störeinflüsse, Ausfälle und mögliche Fehlbedienungen sowie deren Auswirkungen angegeben.

Störungsursache	Störungsauswirkung
Störeinflüsse:	Folgen der Störeinflüsse:
- Wasserdruck in der Zuleitung unter 1,5 atü	- sinkender Wasserstand im Dampfkessel
- Auslaufsieb verkalkt	- Wasserauslauf zu langsam
- kein Mahlgut im Vorratsbehälter	- Dauerlauf der Mühle
- Mühlen-Überstromschutz ausgelöst	- Mühle läuft nicht
Ausfälle:	Folgen der Ausfälle:
- Temperaturreglung des Wasserkessels	- Abfall bzw. Anstieg der Temperatur; bei Ansprechen des Sicherheitsventils muß die Anlage sofort ausgeschaltet werden.
- Druckregelung des Dampfkessels	- Abfall bzw. Anstieg des Druckes; bei Ansprechen des Sicherheitsventils muß die Anlage sofort ausgeschaltet werden.
- Magnetventil(e)	- kein Wasserzulauf in die Kessel
Fehlbedienungen:	Folgen der Fehlbedienungen:
- Wasserabsperrhahn nicht geöffnet	- kein Wasserzulauf in die Kessel; bei einer Temperaturzunahme auf 150°C werden die Heizungen durch das Ansprechen des betreffenden Trockengehschutzes abgeschaltet
- Hand-AUS-Schalter der Mühle betätigt	- statt Kaffee läuft Wasser aus (Mühle läuft nicht)
- Sperrkontakt offen	- keine Kaffee-Entnahme möglich (Maschine nicht "bereit")
- Schlüsselschalter AUS	- " -

Bild 3.1o: Irreguläre Betriebsläufe

4. Gliederung der Automatisierungsaufgaben bei der Betriebsüberwachung von Fließprozessen in Teilaufgaben

Im letzten Abschnitt wurden die Anforderungen an ein Modell dieses Prozeßtyps mit der Aufgabenstellung Betriebsüberwachung erarbeitet.

Durch die Beschreibung des Modells, seiner Variablen, der regulären und irregulären Prozeßzustände lassen sich die Funktionen der Betriebsüberwachung auf den Modellprozeß anwenden.
Bei der Betrachtung der Aufgabenstellung zur Ermittlung ihrer Anforderungen an eine Sprache bzw. zur Beurteilung von Sprachelementen einer vorliegenden Sprache gibt es zwei mögliche Arten des Vorgehens:

1. "Willkürliches" Herausgreifen beispielhafter vollständiger Überwachungsaufgaben am Modellprozeß

2. Aufteilung der Lösungsschritte für die Funktionen der Betriebsüberwachung in typische Teilaufgaben und in deren untergeordnete Aufgabenelemente.

   Die Lösungsschritte können dabei größtenteils am Modellprozeß abgeleitet, teils der Literatur [22], [26] sowie Programmpaketbeschreibungen [27], [28] und [29] entnommen werden.

Die erste Methode hat den Nachteil, daß eine vollständige Überwachungsaufgabe sehr unterschiedliche Anforderungen an eine Programmiersprache stellt und daher zu ihrer Lösung verschiedenartige Sprachelemente erfordert.

Dieser Nachteil wird beim zweiten Weg durch die systematische Vorgehensweise weitgehend vermieden. Durch die Aufgliederung in Teilaufgaben bzw. Aufgabenelemente wird erreicht, konkrete Aussagen über Sprachanforderungen bzw. Sprachelemente für einzelne charakteristische Aspekte der Aufgabenstellung Betriebsüberwachung machen zu können, aus deren Zusammenfassung die Gesamtbeurteilung der zu untersuchenden Programmiersprache folgt.

Für das aus diesen Gründen bevorzugte zweite Verfahren werden die Lösungsschritte in folgende sieben Teilaufgaben gegliedert:

-1 Datenmodellerstellung

-2 Datenmodell-Dokumentation/Modifikation

-3 Prozeßdatenerfassung

-4 Prozeßdatenverarbeitung

-5 Protokollierung

-6 Steuerung von Automatisierungsaufgaben

-7 Synchronisierung von Automatisierungsaufgaben.

Der Hauptgesichtspunkt bei der Aufteilung war aus oben erwähnten Gründen derjenige, daß innerhalb einer Teilaufgabe bzw. eines Aufgabenelements möglichst nur gleichartige Sprachelemente auftreten sollten.
Die obengenannten sieben Teilaufgaben lassen sich in drei Bereiche einteilen:

- Durchführung von Automatisierungsaufgaben

  Hierunter wird die Ausführung von Aufgaben verstanden, ohne Mitberücksichtigung ihrer Steuerung bzw. Koordinierung mit anderen Aufgaben (Teilaufgabe 1, 2, 3, 4 und 5).

- Steuerung von Automatisierungsaufgaben

  Hierbei ist der Start bzw. die Beendigung von Aufgaben zu verstehen, ohne Mitberücksichtigung des zeitlichen Ablaufs anderer Aufgaben. Sowohl der Start als auch die Beendigung können dabei an Zeitbedingungen bzw. an Prozeßzustandsänderungen geknüpft sein (Teilaufgabe 6).

- Synchronisierung von Automatisierungsaufgaben

  Unter diesem Begriff ist die Anpassung des Ablaufs einer Aufgabe an den Ablauf anderer Aufgaben unter Einhaltung vorgegebener Bedingungen zu verstehen (Teilaufgabe 7).

Diese Teilaufgaben bilden die Grundlage für die Sprachanforderungen bzw. der Sprachuntersuchung. Aus diesem Grund sollen sie anhand typischer Beispiele beim Modellprozeß erläutert werden.

4.1 Teilaufgabe 1: Datenmodellerstellung

Unter der Bezeichnung Datenmodell sei hier die geordnete Menge von Datenelementen verstanden. Sie beinhalten

- Werte von Prozeßgrößen
- Werte, die zur Bearbeitung der Automatisierungsaufgaben notwendig sind, z.B. Grenzwerte.

Einfluß auf Zusammensetzung und Aufbau des Datenmodells haben

- Automatisierungsaufgaben und deren zeitliche Steuerung bzw. Koordinierung
- Prozeß (z.B. Anzahl von Prozeßgrößen)
- Meßkanal (z.B. Berücksichtigung von Vorverstärkern)
- Prozeßrechner (z.B. kleinste Zykluszeit).

Die Datenmodellerstellung besteht aus der Datenmodellvereinbarung und der Datenmodellinitialisierung.

4.1.1 Datenmodellvereinbarung

Die Vereinbarung von Daten umfaßt folgende Angaben:

- Bezeichnung der Daten mit Namen (Identifikation)
- Gültigkeitsbereich der Namen; diese können beispielsweise nur in einem Aufgabenelement oder nur innerhalb einer Teilaufgabe bekannt sein.
- Zuweisungsschutz
  - Konstante: lesender Zugriff
  - Variable : lesender und schreibender Zugriff, d.h. keine Zugriffsbeschränkung
- Datentyp einschließlich Genauigkeit: Hierbei werden die Charakteristiken von Datenelementen mit Attributen festgelegt, beispielsweise für eine ganze bzw. rationale Zahl, Bitkette, Zeichenkette usw.
- Speicherresidenz: Daten, auf die mehrere Tasks rasch und häufig zugreifen müssen - dies ist bei dieser Aufgabenstellung sehr oft der Fall - werden aus Zugriffszeitgründen hauptspeicherresident abgelegt.

- Datengruppierung
  - Einfaches Datenelement eines bestimmten Datentyps
  - Bereich: n-dimensionale, geordnete Menge von Datenelementen desselben Typs
  - Struktur: Anordnung von einfachen Datenelementen, Bereichen und Strukturen; die Strukturelemente brauchen in ihren Attributen nicht übereinzustimmen.

Im Zusammenhang mit Bereichen und Strukturen wird die Frage nach der Nutzung des Hauptspeichers akut. Durch die Forderung nach Speicherresidenz kann bei Vorhandensein vieler Prozeßvariablen bzw. zahlreicher Automatisierungsaufgaben der Hauptspeicherbedarf des zentralen "Überwachungsrechners" sehr groß werden [31] .
Der Anwender fordert, zwischen den beiden Alternativen wählen zu können (Untersuchung des Aufwands (Kosten) siehe [3o]):

1. Optimale Speicherausnutzung

   Die Datenelemente werden zusammenhängend abgelegt. Durch das Packen und Entpacken der Datenelemente während der Ausführung des Programms wird die Rechenzeit entsprechend länger.
   Beispiel: Datenvereinbarung in PL/1 mit dem Attribut UNALIGNED.

2. Optimale Zugriffs- und Verarbeitungszeit

   Das Ablegen der Datenelemente erfolgt auf Wort-Halbwort-Byte-grenze, d.h. auf Speichergrenzen.

Die Forderung nach optimaler Speicherausnutzung verliert durch die sinkenden Speicherkosten ebenso an Bedeutung wie durch die zunehmende Dezentralisierung von Automatisierungssystemen, d.h. der Verteilung von Aufgaben auf verschiedene Rechner.

Beispiel einer Datenvereinbarung in PEARL:

```
ERFASSUNG: TASK;
DECLARE BITFELD(8) BIT(1) RESIDENT;
:
END;  /* TASKENDE */
```

Kennzeichen der Vereinbarung:	Schlüsselwort DECLARE (Abk.: DCL)
Bezeichner (Namen):	BITFELD
Gültigkeitsbereich:	Task ERFASSUNG
Zuweisungsschutz:	nein
Datentyp:	BIT (Bitkette)
Genauigkeit:	1 (Bitkette der Länge 1)

Speicherresidenz: ja (Attribut RESIDENT)
Datengruppierung: 8 Datenelemente.

Die Ablage der einzelnen Datenelemente kann in PEARL vom Anwender nicht beeinflußt werden. Ob sie bei diesem Beispiel in einem oder in 8 Bytes bzw. Worten erfolgt, ist implementationsabhängig.

4.1.2 Datenmodellinitialisierung

Datenmodellinitialisierung wird hier bezeichnet als das Vorbesetzen der Datenelemente mit Anfangswerten vor dem Beginn der eigentlichen Überwachung. Die Anfangswerte dürfen dabei - je nach den Zugriffsrechten - während der Betriebsüberwachung geändert werden oder auch nicht:

- Initialisierung konstanter Größen
- Initialisierung variabler Größen.

Abhängig vom Datenmodell bzw. der Überwachungsaufgabe erfolgt die Initialisierung mit unterschiedlichen Sprachmitteln:

- Anfangswertvereinbarung

  In diesem Fall werden konstante und variable Größen bereits bei der Vereinbarung vorbesetzt.

  - Konstante
    Beispiele hierfür sind Linearisierungskoeffizienten, Meßstellennamen usw.
  - Variable
    Mit einer Anfangswertvereinbarung werden Variable initialisiert, die zu Beginn der Überwachung bestimmte Werte besitzen, die sich aber evtl. ändern können. Ein Beispiel hierfür sind Umrechnungskoeffizienten:
    DECLARE KOEFFIZIENT FLOAT INITIAL (1.1);

- Eingabeanweisung

  Hiermit werden Variable vorbesetzt, deren Anfangswerte man häufig ändern möchte, z.B. Grenzwerte, Zykluszeiten usw.

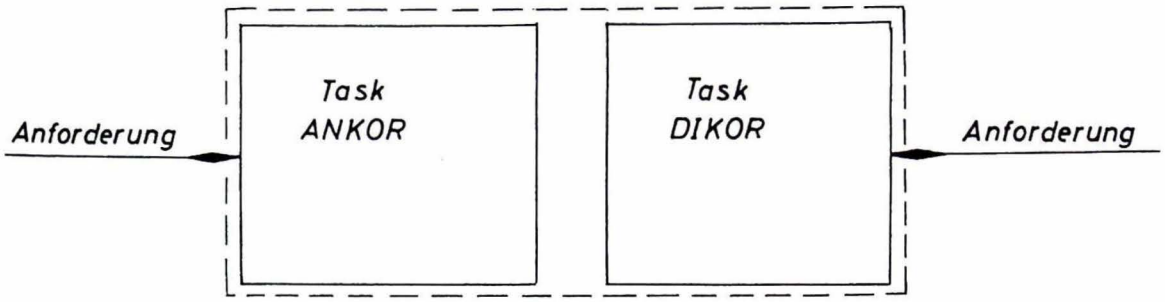
- Zuweisungs-Anweisung

  Mit einer Zuweisung werden Variable initialisiert, die abhängig si von
  - Prozeßzustand zur Zeit des Überwachungsbeginns (z.B. Anfangstemperatur im Wasserkessel)
  - Eingabedaten.

4.2 Teilaufgabe 2: Datenmodelldokumentation - Modifikation

Unter diesen Begriffen versteht man das Dokumentieren und/oder Modifizieren einzelner Datenmodellkomponenten einer anzugebenden Prozeßgröße über die Prozeßbedienperipherie. Neben der reinen Dokumentation ist es insbesondere bei der Inbetriebnahme von Automatisierungsprogrammen notwendig, sich einzelne Komponenten protokollieren lassen zu können und sie gegebenenfalls zu ändern.

Das Blockdiagramm des Programmsystems für die Dokumentation und Modifikation beim Modellprozeß [32] hat folgendes Aussehen:



Abhängig davon, ob Werte von Analoggrößen oder Binärgrößen dokumentiert bzw. modifiziert werden sollen, werden die Tasks ANKOR oder DIKOR von der Bedienungsschreibmaschine aus aktiviert.

Die Aufgaben der Tasks sind im einzelnen:

Dokumentation

- Anwahl der Dokumentationsfunktion
- Eingabe des Prozeßgrößennamens
- Eingabe der Komponente, z.B. Grenzwertdokumentation
- Dokumentation, d.h. Ausdruck auf Drucker oder Blattschreiber.

Modifikation

- Anwahl der Modifikationsfunktion
- Eingabe des Prozeßgrößennamens
- Eingabe der zu modifizierenden Komponente
- Prüfung, ob die Komponente vom Bedienungspersonal modifiziert werden darf
- Eingabe des neuen Wertes
- Prüfung des Wertes auf Plausibilität; wenn der Wert gültig ist, erfolgt die Abspeicherung des Wertes und die Übernahmequittierung.

Beispielhafter Ablauf der Task ANKOR:

```
DATENMODELLDOKUMENTATION UND -MODIFIKATION   : Programmüberschrift
FUER ANALOGGROESSEN
KURZNAME ?
INPUT     ANKOR : TDK2                       : Anforderung des Meßstellen-
NR.  2    DAMPFKESSELTEMP.2                    kurznamens
FUNKTION ?
INPUT     ANKOR : DOK                        : Anforderung der Funktion
KOMPONENTE ?                                   (hier DOKUMENTATION)
INPUT     ANKOR : PLAU                       : Anforderung der Komponente
UNTERE PLAUS.GRENZE =  20.00 GRDC              (hier Plausibilität)
OBERE  PLAUS.GRENZE = 160.00 GRDC            : Dokumentation der Werte
FUNKTION ?
INPUT     ANKOR : MOD                        : Anforderung der Funktion
KOMPONENTE ?                                   (hier MODIFIKATION)
INPUT     ANKOR : GREW                       : Anforderung der Komponente
UNTERER GRENZWERT ?                            (hier Grenzwert)
INPUT     ANKOR : 15.00                      : Anforderung d. unteren Grenze
ZU NIEDER                                    : Falsche Eingabe
UNTERER GRENZWERT ?
INPUT     ANKOR : 120.00                     : Erneute Anforderung
OBERER  GRENZWERT ?
INPUT     ANKOR : 90.00                      : Anforderung der oberen Grenze
UNTERER GRENZWERT >= OBERER  GRENZWER        : Grenzwerte stimmen nicht übere
UNTERER GRENZWERT ?
INPUT     ANKOR : 90.00                      : Erneute Anforderung
OBERER  GRENZWERT ?
INPUT     ANKOR : 130.00
BESTAETIGUNG !
INPUT     ANKOR : 0                          : Bestätigung der Eingaben
UEBERNOMMEN    2. 2.1977  12:31:10           : Übernahmebestätigung
```

4.3 Teilaufgabe 3: Prozeßdatenerfassung

Voraussetzung der Überwachung von Prozeßdaten mit einem Prozeßrechner ist das Bereitstellen ihrer Werte in der Zentraleinheit; unter Prozeßdatenerfassung wird dabei hier allein das Erfassen der Werte der Prozeßgrößen verstanden, ohne anschließende Aufbereitung bzw. Weiterverarbeitung. Das bedeutet bei analogen Prozeßdaten das Abspeichern der Rohwerte $x_{iR}(t_k)$, bei Binärgrößen das Abspeichern einzelner bzw. gruppenweise vorliegender Binärwerte $x_i(t_k)$ (t_k: Zeitpunkt der Abfrage).
Die Erfassung wird also aufgefaßt als Anweisung zur Datenübertragung von der physikalischen Endstelle (Datenquelle) zur logischen Endstelle (Datensenke).
Zu unterscheiden ist zwischen der Eingabe von Analog- und Binärgrößen über die Funktionseinheiten Analogeingabe und Digitaleingabe (s. DIN 66216).

4.3.1 Analogeingabe

Über diese Einheit werden amplitudenanaloge Signale erfaßt.
Dabei können folgende Angaben erforderlich werden:

- Einlesemodus (Eingabe über ein Arbeitsregister bzw. direkt in den Speicher, evtl. Prozessorfreigabe während der Eingabe)
- Verstärkungsfaktor
- Integrationszeit
- Hardware - Filter
- Gerätefehler - Reaktion.

4.3.2 Digitaleingabe

Diese Einheit dient der Erfassung der Werte von

- Binärgrößen (einzeln oder gruppenweise vorliegend)
- Analoggrößen (bei digitaler bzw. frequenzanaloger Signaldarstellung).

In diesem Fall ist anzugeben, ob die Daten bitweise, wortweise oder blockweise (DIN 66216) eingelesen werden.

4.4 Teilaufgabe 4: Prozeßdatenverarbeitung

Diese Teilaufgabe schließt sich an die Erfassung einer bzw. mehrerer Prozeßgrößen an. Sie umfaßt mehrere typische Aufgabenelemente, die im Rahmen der Sprachuntersuchung programmiert und daher zusammenfassend erläutert werden sollen.

4.4.1 Linearisierung von Fühlerkennlinien

Teils prinzipbedingt, teils aufgrund der Verbesserung der Fühlereigenschaften erfolgt die Umformung physikalischer Größen in elektrische Signale sehr häufig nichtlinear. Die verwendeten Methoden der Kennlinienlinearisierung über Programme sind folgende:

1. Verwendung mathematischer Funktionen

   Die Meßwerte werden über eine Funktion korrigiert, die vom Anwender aufgrund von Messungen (Approximation [33] , [34]) oder vom Fühlerhersteller bzw. der DIN-Norm angegeben wird.

Beispiel: Widerstands-Temperaturgeber Pt 100 zur Messung der Temperaturen in den Kesseln

$$R_\vartheta = R_0 \, (1 + A \cdot \vartheta + B \cdot \vartheta^2)$$

R_ϑ - Widerstand bei Temperatur ϑ

R_0 - Widerstand bei $0^\circ C$

ϑ - Temperatur in $^\circ C$

A - $0.39078\ 10^{-2}\ grd^{-1}$

B - $0.578\ 10^{-6}\ grd^{-2}$

} DIN-Angaben

2. Stützpunktmethode

Die frei wählbaren Stützpunkte der Umrechnungskurve werden punktweise abgespeichert.

Dem Vorteil der ersten Methode - geringer Speicherplatz - steht der Vorteil der zweiten Methode - geringe Rechenzeit - gegenüber.

4.4.2 Digitale Filterung

Zur Minimierung des Fehlers zwischen Nutzsignal und gestörtem Signal ist die digitale Filterung ein weiteres Verfahren, neben den bekannte Maßnahmen wie z.B. Verdrillung, Erdung der Leitungen. Sie kann als softwaremäßige Korrektur eines Rohwertes $x_{iR}(t_k)$ bzw. Fertigwertes $x_{ip}(t_k)$ nach einem zu wählenden Algorithmus [35], [37] angesehen werden. Zur Unterscheidung von der Glättung soll die Filterung so verstanden werden, daß nur einige zurückliegende Meßwerte bzw. der aktuelle Meßwert berücksichtigt wird, während bei der Glättung eine aus einer vollständig vorliegenden Meßreihe resultierende Kurve überall "glatt" sein soll [36].

Beispiel: Filterung der zyklisch erfaßten Analoggröße "Wasserkesseldruck" mit einem Filter 1. Ordnung (Berücksichtigung eines zurückliegenden Meßwertes). Berechnung:

$$x_{11RF}(kT) = p x_{11RF}\left[(k-1)T\right] + (1-p) \cdot x_{11R}(kT) \quad ; \text{ dabei bedeuten:}$$

x_{11R} : Rohwert (Meßstelle 11)

x_{11RF} : gefilterter Rohwert

kT : bei zyklischer Abfrage ist $t_k = kT$ mit $k = 1,2,3 \ldots \infty$ und T als Zyklusdauer

p : Dämpfungsfaktor

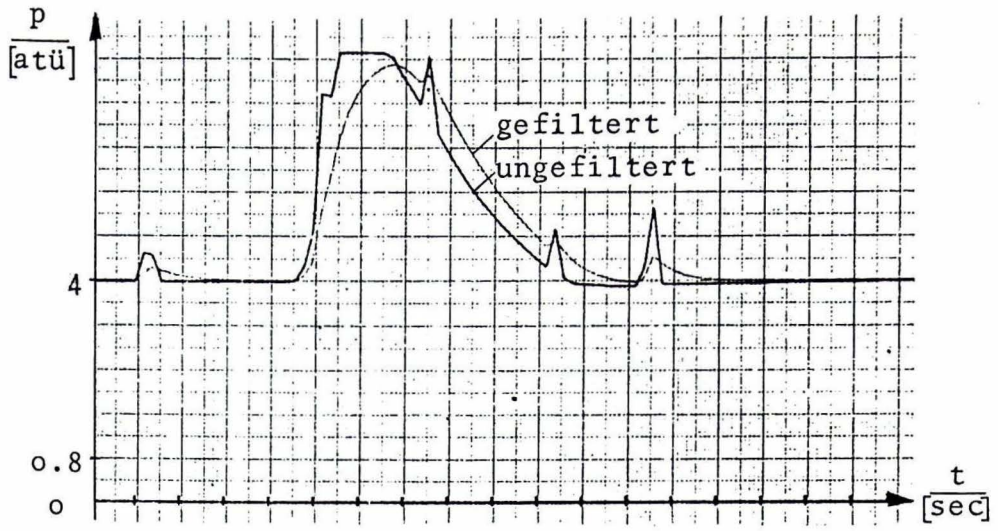


Bild 4.1: Druckverlauf im Wasserkessel mit/ohne Filterung

4.4.3 Umrechnung von Rohwerten in ihre physikalischen Einheiten

Mit einer linearen Funktion+) wird der Rohwert x_{iR} in den der physikalischen Größe entsprechenden Zahlenwert x_{ip} umgerechnet. Diese Umrechnung wird auch als Zahlenanpassung bezeichnet.
Beispiel: Berechnung der Umrechnungskoeffizienten des Wasserkesseldrucks:

$$x_{ip} = A + B \cdot x_{iR}$$

Messung I $\begin{cases} x_{11R} = 1000 \\ x_{11p} = 10 \ [\text{atü}] \end{cases}$ $A = 0$

Messung II $\begin{cases} x_{11R} = 0 \\ x_{11p} = 0 \ [\text{atü}] \end{cases}$ $B = \frac{10}{1000} = 0{,}01$

4.4.4 Fertigwertkorrektur

Unter Fertigwertkorrektur sei hier die Modifikation eines Fertigwertes durch eine bestimmte Korrekturfunktion zu verstehen. Sie ist dann notwendig, wenn ein Zusammenhang zwischen zwei oder mehreren Meßgrößen festzustellen ist und daher bei der Berechnung eines Fertigwertes andere Meßgrößen (Korrekturgrößen) hinzugezogen werden müssen.
Die Korrekturfunktion kann dabei aufgrund eines analytischen Modells oder auch aus Meßwerten mit Hilfe der Methode der kleinsten Quadrate (Regressionsanalyse) experimentell ermittelt werden.

+) Die Gebernichtlinearität wird bereits bei der Fühlerlinearisierung berücksichtigt.

Beispiel:
Der Fertigwert x_{27p} des Dampfkesselfüllstands muß beim Modellprozeß durch Mitberücksichtigung der Dampfkesseltemperatur x_{4p} über die folgende Funktion korrigiert werden:

$x_{27pk} = x_{27p} + 0.0178 \cdot x_{4p} - 0.0005 \cdot x_{4p}^2$ (x_{27pk}: korrigierter Fertigwe

4.4.5 Gültigkeitsprüfung (Plausibilitätsprüfung)

Zur Unterscheidung, ob bei Überschreiten gewisser Grenzwerte bzw. bei Vorliegen gewisser binärer Werte der technische Prozeß oder aber die Meßanordnung gestört ist, werden folgende programmtechnische Maßnahme getroffen:

1. Statische Gültigkeitsprüfung

   Analoggrößen können auch in Störungsfällen nicht beliebige Werte annehmen, d.h. sie können auf einen festen unteren und/oder oberen Grenzwert geprüft werden.
   Beispiel:
   Die obere Gültigkeitsgrenze für den Durchfluß beträgt 0,5 l/min. Größere Mengen sind durch eine eingebaute Durchflußblende nicht möglich.

2. Dynamische Gültigkeitsprüfung (Gradientenprüfung)

   Analogwerte können sich nur mit einer begrenzten Geschwindigkeit ändern. Es läßt sich also eine maximale Meßwertdifferenz zwischen zwei Abtastungen angeben, die im regulären und auch gestörten Betrieb nicht überschritten wird. So ist beispielsweise der Gradien des Dampfkesselfüllstands sowohl beim Wasserzulauf alsauch bei der Entnahme auf bestimmte Werte begrenzt.

3. Sekundärwertprüfung

   Im allgemeinen ist eine Prozeßgröße selten unabhängig von anderen Größen. Diese Tatsache macht man sich hier zunutze, indem man bei Verletzen eines Grenzwertes (bei Analoggrößen) bzw. eines Zustandes (bei Binärgrößen) noch mindestens eine andere Größe (Sekundärgröße) einer Prüfung unterzieht. Zeigt auch diese Größe eine entsprechende Abweichung, so ist daraus zu schließen, daß der Prozeß gestört ist, im anderen Fall kann angenommen werden, daß die Meßanordnung defekt ist. Ein Beispiel hierfür ist die Überprüfung der Primärgröße 'Dampfkessel-Wassertemperatur' durch die Sekundärgröße 'Dampfkesseldruck'.

4.4.6 Grenzwertprüfung

1. Prüfung von Analogwerten auf untere und/oder obere Grenzwerte

Sehr häufig lassen sich für zeitkontinuierliche Analoggrößen Grenzwerte angeben, die im ungestörten Betrieb nicht verletzt werden dürfen. Zur Kennzeichnung des Grades der Verletzung werden i.a. mehrere Grenzen eingeführt.
Beispiel: Wasserkesseldrucküberwachung

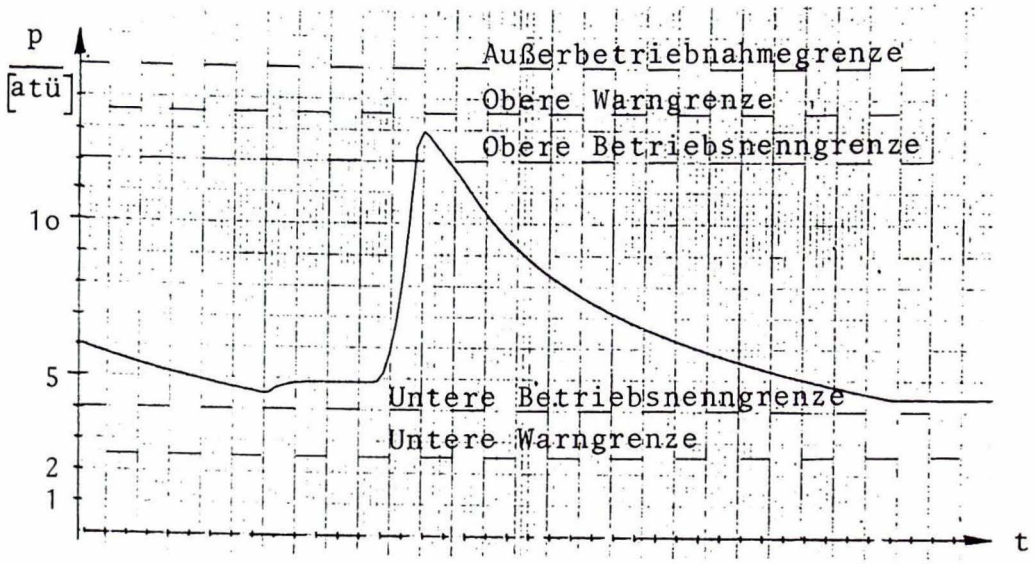


Bild 4.2: Grenzen für den Wasserkesseldruck

Die einzelnen Grenzen haben folgende Bedeutung:
Betriebsnenngrenze: Kurzzeitige Überschreitung zulässig
Warngrenze: Unzulässiger Betriebszustand (Störungsmeldung)
Außerbetriebnahmegrenze: (Not-) Abschalten des Prozesses erforderlich, Gefahr der Kesselzerstörung.

Zu unterscheiden ist zwischen zwei Arten von Grenzen, nämlich

- unabhängigen Grenzwerten ("feste" Grenzen)
- abhängigen Grenzwerten ("gleitende" Grenzen)
  - zeitabhängige ("dynamische") Grenzen
  - prozeßgrößenabhängige Grenzen.

2. Prüfung auf maximale Änderungsgeschwindigkeit (Tendenzüberwachung)

Der Quotient von Meßwertdifferenz und der Zeit zwischen zwei Abtastungen wird auf einen konstanten (variablen) Grenzwert überwacht:

$$\frac{\left|x_{ip}(kT) - x_{ip}\left[(k-1)T\right]\right|}{T} \leqq \left|\frac{\triangle x_{ip}}{T}\right|_{max}$$

Beispiel: Überwachung der Temperaturzunahme im Wasserkessel bei eingeschalteter Heizung. Bei zu großem Temperaturgradient muß folgende Störungsmeldung erfolgen+):
ZU NIEDRIGER WASSERSTAND, WASSERZULAUF ÜBERPRÜFEN

4.4.7 Kennwertberechnung

Es ist häufig notwendig, aus Analogwerten über bestimmte Verknüpfungen sogenannte Kenngrößen abzuleiten. Gründe hierfür sind

- Datenreduktion
- "Verdichtung" zu charakteristischen Größen, die i.a. aussagekräftiger sind als die Meßgrößen selbst.
  Beispiel: Räumlicher Temperaturmittelwert (x_{Wkp}) im Wasserkessel:

$$x_{Wkp} = \frac{x_{5p} + x_{6p} + x_{7p} + x_{8p} + x_{9p}}{5}$$

4.4.8 Binärwertverknüpfungen

Binärwerte werden mittels logischer Operatoren wie NOT, AND, OR, EXOR [38] verknüpft. Beispiel: Feststellen des Prozeßzustandes BEREIT anhand von verschiedenen Binärwerten B1 ... B4:

B1 "Sperrkontakt" GESCHLOSSEN
B2 "Lichtschranke Glasbehälter" GESCHLOSSEN
B3 "Lichtschranke Kaffee-Entnahmestelle" GESCHLOSSEN
B4 "Schlüsselschalter" GESCHLOSSEN

Formulierung dieser Aufgabenstellung als Boole'sche Gleichung:

$$BEREIT = B1 \wedge B2 \wedge B3 \wedge B4$$

+) Der Heizungsstrom wird als korrekt vorausgesetzt

4.5 Teilaufgabe 5: Protokollierung

Zur Information des Bedienungspersonals über die erfaßten bzw. verarbeiteten Werte werden diese in schriftlicher, vom Menschen lesbarer Form ausgegeben. Protokolle enthalten normalerweise Datum, Uhrzeit, Protokollidentifikation, Meßstellennamen, Wert der Prozeßgröße mit Dimension, Abweichung von Sollwerten, Texte sowie den Ausgabeort. Folgende Angaben sind also notwendig:

1. Ausgabedaten (Datenmodellkomponenten, Kommentare)
2. Ausgabeformat
3. Ausgabegerät(e).

Man unterscheidet grundsätzlich zwischen dem Betriebsprotokoll und dem Störungsprotokoll.

4.5.1 Betriebsprotokoll

Betriebsprotokollierung heißt, das Betriebspersonal in regelmäßigen Zeitabständen (zyklisch) über das Prozeßgeschehen zu informieren. Dazu werden die Werte der wichtigsten Prozeßgrößen bzw. Kenngrößen dokumentiert.

4.5.2 Störungsprotokoll

Dieses Protokoll wird dem Bedienungspersonal spontan als Information über eine aufgetretene Störung ausgegeben. Diese kann sowohl aus dem Prozeß als auch aus dem Bereich der Meßeinrichtung herrühren. Während im zweiten Fall dokumentiert wird, welcher Meßkanal defekt ist, gibt es im ersten Fall zwei Arten von Meldungen:

- Störungsmeldungen ohne Angabe der Störungsursache. Es werden nur die Werte der gestörten Größen bzw. deren Abweichungen von Sollwerten protokolliert. Sowohl die Störungsdiagnose als auch die Reaktion auf die Störung bleibt dem Bedienungspersonal überlassen.
  Beispiel:
  15.7.1976 12:48:34 UHR
  DAMPFKESSELTEMPERATUR 148.2 GRDC, ZU HOCH; GRENZWERT: 140 GRDC
- Störungsmeldungen mit Angabe der Störungsursache.
  Zusätzlich zu obigen Mitteilungen wird dokumentiert, welche Baueinheit defekt ist bzw. wie das Bedienungspersonal auf die Störung zu reagieren hat.

4.6 Teilaufgabe 6: Steuerung von Automatisierungsaufgaben

4.6.1 Steuerung der Prozeßdatenerfassung

Abhängig vom zeitlichen Verlauf der Prozeßgrößen sowie den zeitlichen Anforderungen an die Erfassung werden die Prozeßvariablen zyklisch oder azyklisch eingelesen.
Die Lösung dieser Problemstellung umfaßt drei Punkte:

- Formulierung der Steuerungsaufgabe
- Mitteilung dieser Aufgabe an ein geeignetes Betriebssystem (im folgenden Einplanung, bei zyklischer Auftragserteilung kurz Zykluseinplanung genannt)
- Löschen der Einplanung.

Zur Formulierung der Steuerungsaufgabe werden vier Komponenten benötigt

- Steuerbedingungen

  Man unterscheidet zwischen drei Typen von Steuerbedingungen:
  - Erreichen einer absoluten Uhrzeit
  - Ablauf einer Zeitdauer
  - Eintreffen eines Unterbrechungssignals (asynchrones Prozeßereignis)

  Dementsprechend lassen sich Parameter der Steuerbedingungen angeben:
  - Uhrzeitwert
  - Zeitdauerwert
  - Bezeichner für ein Unterbrechungssignal.

- Steueroperation

  Steueroperation bedeutet Steuerfunktion im Sinne von Anstoß, Unterbrechung, Fortsetzung und Beendigung der Bearbeitung einer Aufgabe. Bei der Prozeßdatenerfassung handelt es sich um das Einleiten einer Tätigkeit, die Steuerbedingungen treten in diesem Zusammenhang nur mit dieser Operation auf.

- Auftragnehmer

  Auftragnehmer sind Objekte, auf welche die Steueroperationen bei Erfüllen von Steuerbedingungen angewandt werden, hier Automatisierun programme zur Durchführung der Prozeßdatenerfassung.

- Ausführungsrandbedingungen (Pünktlichkeit der Operationsausführung)

  Es handelt sich hierbei um Angaben, inwieweit Abweichungen vom vorge schriebenen Erfassungszeitpunkt zulässig sind, wie beispielsweise di maximale Zeitdifferenz zwischen dem Erfüllen der Bedingung und der tatsächlichen Erfassung.

Während die Punkte Steueroperation, Auftragnehmer und Ausführungsrandbedingungen dadurch weitgehend formuliert sind, ist die Beschreibung der Steuerbedingungen sehr viel umfangreicher. Daher soll nun auf die wichtigsten Bedingungen für den Anstoß der Prozeßdatenerfassung eingegangen werden.

Zyklische Erfassung

Der Begriff zyklische Erfassung ist im folgenden zu verstehen als Abtasten von Prozeßgrößen in äquidistanten Zeitpunkten.
Die zyklische Erfassung ist beim vorliegenden Prozeßtyp durch den zumindest stückweise kontinuierlichen Verlauf der Prozeßvariablen kennzeichnend.

Prozeßvariable, die bei Vorliegen anzugebender Prozeßzustände bzw. Uhrzeiten durch ständige, regelmäßige Abfrage überwacht werden können, werden zyklisch eingelesen.

Ein Zyklus erfordert zu seiner Beschreibung mindestens drei Angaben:

- Angaben über Bedingungen, bei deren Erfüllen die erste Erfassung erfolgt: Zyklusstart
- Angaben über Wiederholungsbedingungen (Zykluszeiten)
- Angaben über Bedingungen, wann die zyklische Erfassung beendet werden muß: Zyklusende.

Die Angaben müssen bzw. können teils explizit, teils implizit erfolgen. Implizit bedeutet, daß eine Bedingung nicht ausdrücklich formuliert wird, explizit bedeutet dementsprechend, daß eine Bedingung ausdrücklich formuliert wird.
Beispiel: Wird keine Startbedingung explizit formuliert, wird der Zyklus unmittelbar nach der Einplanung gestartet.

Zu erwähnen ist, daß auch implizit angegebene Start- bzw. Endebedingungen im folgenden als Start- bzw. Endebedingungen bezeichnet werden.

Die Steuerbedingungen für die Organisation der zyklischen Auftragserteilung lassen sich wie folgt einteilen:

a) Zyklus-Startbedingungen

Startbedingungen sind Bedingungen für das erstmalige Ausführen einer Operation. Nach Erfüllen einer der nachstehenden Bedingungen erfolgt der Zyklusbeginn, wobei zwei Fälle zu unterscheiden sind:

-1 Die Zykluseinplanung und der Zyklusstart sind zeitlich gekoppelt, die Einplanung impliziert den Zyklusbeginn ("impliziter Zyklusbeginn").
Beispiel:
Beim Programmsystem für den Modellprozeß werden die verschiedenen Zyklen nach dem Einlesen der meßstellenspezifischen Informationen eingeplant und sofort gestartet.

-2 Die Zykluseinplanung und der Zyklusstart sind zeitlich entkoppelt, die Startbedingungen müssen explizit formuliert werden ("expliziter Zyklusbeginn").

Folgende Startbedingungen treten auf:

- Erreichen einer absoluten Uhrzeit
  Beispiel:
  Die Überwachung des Modellprozesses auf den Betriebszustand "bereit" ist erst nach Erreichen einer bestimmten Uhrzeit notwendig.

- Ablauf einer Zeitdauer
  Beispiel:
  Zyklische Überwachung der Arbeitsphase des Modellprozesses nach dem Einschalten der Anlage.
  Die Lösung dieser Aufgabe sieht folgendermaßen aus:
  Nachdem anhand des Zustandes der Prozeßgröße HAUPTSCHALTER festgestellt worden ist, daß die Maschine eingeschaltet ist, läßt sich die zyklische Überwachung der Arbeitsphase durch einen verzögerten Zyklusstart einfach einplanen. Die Verzögerungszeit entspricht der bekannten Zeitdauer der Anlaufphase.

- Eintreffen eines Unterbrechungssignals

  - Der Start einer zyklischen Taskbeauftragung wird von einem definierten Unterbrechungssignal eingeleitet.
    Beispiel:
    Zyklische Erfassung des Heizungsstromes nach Einschalten der Wasserkesselheizung

- Der Start einer zyklischen Beauftragung wird von einem Unterbrechungssignal - bei mehreren für den Start vorgesehenen Unterbrechungssignalen - eingeleitet.
  Beispiel:
  Zyklische Erfassung des Durchflusses bei Anziehen eines der beiden Magnetventile.
  Der Auftragnehmer benötigt in diesem Fall die Kenntnis des Auftraggebers, so ist es beispielsweise für die Grenzwertüberprüfung erforderlich, zu wissen, welches der beiden Magnetventile angezogen hat.

Der Zyklusstart nach dem Eintreffen eines Interrupts kann um eine gewisse Zeitdauer verzögert erfolgen, beispielsweise darf die zyklische Erfassung der Pumpenmotordrehzahl nach dem Einschalten des Motors erst nach dessen Hochlaufen erfolgen.

b) Wiederholungsbedingung (Zykluszeit T)

Der Begriff Zykluszeit hat nach DIN 44 300, Nr. 126 folgenden Wortlaut: "Bei einer Funktionseinheit die Zeitspanne zwischen dem Beginn zweier aufeinanderfolgender, zyklisch wiederkehrender Vorgänge".

Als Zykluszeit einer zu überwachenden Binärgröße wird die maximal zulässige Zeitdauer zwischen dem Zeitpunkt des Eintretens einer Binärwertänderung und dem Erkennen der Änderung durch den Prozeßrechner gewählt.

Die Zykluszeit bei Analoggrößen ist abhängig vom Zeitverhalten einer Größe und vom Abstand des Analogwertes von Grenzwerten: je größer die Änderungsgeschwindigkeit bzw. je mehr ein Analogwert in die Nähe einer Grenze (z.B. Meldegrenze) kommt, umso häufiger muß die Analoggröße erfaßt werden.

Aus der Ableitung geht hervor, daß die Zykluszeit grundsätzlich eine variable Größe ist, abhängig vom Prozeßzustand und dem Zeitverlauf der Prozeßvariablen in diesem Zustand. Für die Wahl der Zykluszeit gelten also zwei gegensätzliche Kriterien: Die Zeit T muß einerseits so klein wie möglich sein, um Störungen möglichst rasch erkennen zu können und muß andererseits aus Gründen der Rechnerbelastung so groß wie möglich sein. Der untere Grenzwert von T ergibt sich aus der Forderung, daß bei durchschnittlichem "Lastprofil" [39] ein Auftrag abgearbeitet ist, wenn infolge der Wiederholungsbedingung ein neuer Auftrag erzeugt wird.

c) Zyklus-Endebedingungen

Endebedingungen sind Bedingungen, bei denen eine Operation das letzte Mal ausgeführt werden kann (nicht muß, im Gegensatz zu den Start- bzw. Wiederholungsbedingungen).

Implizites Zyklusende

Implizit bedeutet hier, daß ein Zyklusende potentiell herbeigeführt wird, ohne daß der Benutzer dabei die Absicht der Zyklusbeendigung explizit ausdrückt. Ein Beispiel hierfür ist die Zykluseinplanung mit dem Wunsch, dadurch einen evtl. vorhandenen Zyklus zu beenden bzw. zu ersetzen.

Explizites Zyklusende

Aus noch zu erläuternden Gründen kann es notwendig werden, die zyklische Auftragserteilung an Programme explizit einzustellen. Bezüglich des Zeitpunktes, zu dem das Zyklusende angegeben wird, lassen sich die Anforderungen an die Formulierung der Endebedingungen in zwei Gruppen einteilen:

1. Die Endebedingung ist bei der Einplanung der Operation enthalten und zwar sowohl der Typ als auch der Parameter der Bedingung:

   Die Formulierung der Start- und der Endebedingung ist gekoppelt.

   Man unterscheidet zwischen folgenden Endebedingungen:

   - Ablauf einer Zeitdauer nach dem Zyklusbeginn
     Beispiel: Erfassen eines Meßwerts für eine bestimmte Zeitdauer zum anschließenden Ausgeben auf ein Zeichengerät.
   - Erreichen eines Zeitpunktes
     Beispiel: Die Überwachung des Modellprozesses auf den Zustand "bereit" ist nur bis zu einem bestimmten Zeitpunkt erforderlich.
   - Eintreffen eines Unterbrechungssignals
     Beispiel: Nach Eintreffen des Signals HEIZUNG-EIN wird der Heizungsstrom bis zum Eintreffen des Interrupts HEIZUNG-AUS zyklisch erfaßt.

2. Das Zyklusende wird nach der Einplanung der Operation vorgegeben. Diese Forderung wird dann gestellt, wenn man sich bei der Einplanung auf den Typ und Parameter der Endebedingungen nicht festlegen will oder kann:
   Die Formulierung der Start- und Endebedingung ist in diesem Fall entkoppelt.

Es gibt dabei zwei Möglichkeiten der entkoppelten Zyklusende-Mitteilung an das Betriebssystem:

- Die Mitteilung erfolgt ohne Angabe von Steuerbedingungen, d.h. ein Zyklus wird sofort beendet. Diese Anforderung kann in den folgenden Fällen auftreten:
  - Im Rahmen der Prozeßüberwachung wird eine Prozeß- bzw. Meßkanalstörung festgestellt, die zum Beenden eines Zyklus' führen muß, da eine weitere Überwachung nicht sinnvoll ist.
  - Über die Korrespondenz wird eine "Endetask" gestartet bzw. eine zyklische Auftragserteilung wird über das Bediensystem beendet.
    Beispiel:
    Ein außerplanmäßiges Abschalten der Anlage, z.B. infolge der Wartung, muß zum Beenden des Zyklus' führen.
- Die Mitteilung erfolgt mit Angabe von Steuerbedingungen.

  Einem zyklischen Auftrag, der ohne Endebedingungen formuliert wurde, werden diese zu einem späteren Zeitpunkt hinzugefügt, das Zyklusende wird also nachträglich eingeplant.

d) Zyklusunterbrechungs- bzw. Fortsetzungsbedingungen

Eine weitere zusätzliche Anforderung an die Zyklussteuerung folgt aus dem Vorhandensein verschiedener Betriebsarten, d.h. der Notwendigkeit der Prozeßdatenüberwachung bei verschiedenen Prozeßzuständen.

Aus der Zulässigkeitstabelle (s. Bild 3.9) bzw. bei deren weiterer Unterteilung geht hervor, daß die Überwachung von Prozeßvariablen, die zyklisch erfaßt werden, bei bestimmten Prozeßzuständen nicht sinnvoll ist. Zur Lösung dieser Aufgabe gibt es zwei Möglichkeiten:

1. Die Tasks zur Erfassung (Verarbeitung) werden zyklisch angestoßen, die tatsächliche Erfassung (Verarbeitung) erfolgt entsprechend der Zulässigkeitstabelle.
   Vorteilhaft bei dieser Alternative ist die programmtechnisch einfache Zyklussteuerung. Als Nachteil erweist sich die unnötige zusätzliche Rechnerbelastung, wenn die Überwachung z.Z. nicht erforderlich ist ("Task-Leerlauf").
2. Die Tasks werden nur gemäß der Zulässigkeitstabelle angestoßen. Nachteilig ist, daß je nach den vorhandenen Sprachmitteln die Zyklussteuerung programmtechnisch mehr oder weniger aufwendig ist.

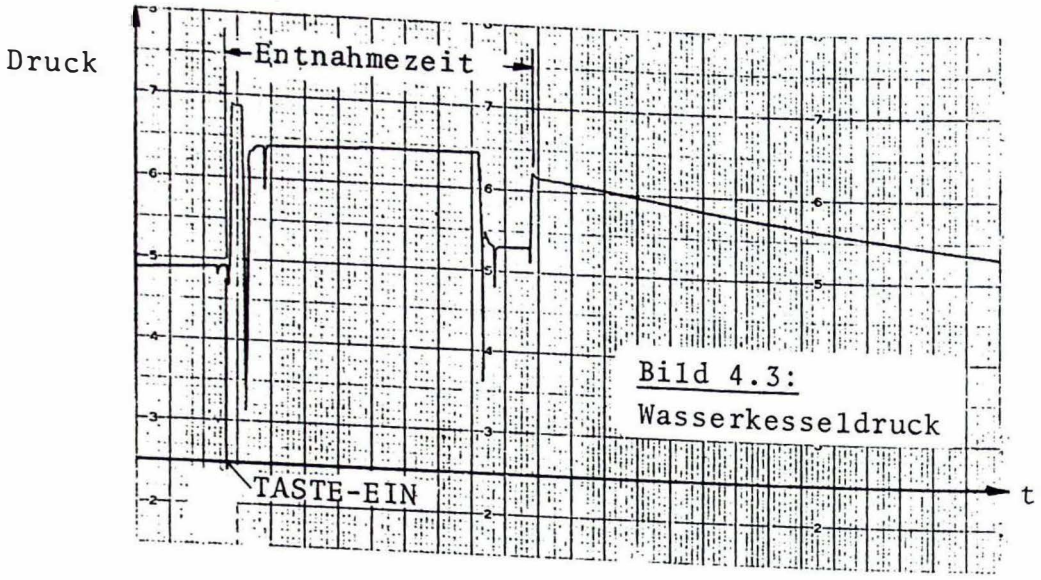


Bild 4.3:
Wasserkesseldruck

Beispiel: Überwachung des Wasserkesseldrucks während der Kaffee-Entnahme (s. Bild 4.3).

Durch die unregelmäßigen Druckspitzen, verursacht vom Öffnen und Schließen der Kesseleinlaß- und Auslaßventile ist eine Drucküberwachung in dieser Betriebsart nicht sinnvoll, der Zyklus muß unterbrochen werden.

Die Zyklusunterbrechungs- bzw. Fortsetzungsbedingungen lassen sich durch die bekannten Ende- und Startbedingungen ausdrücken. Für das Bild 4.3 bedeutet dies:

- Die Unterbrechungsbedingung entspricht der Endebedingung "Eintreffen eines Interrupts"
- Die Fortsetzungsbedingung entspricht der Startbedingung "Ablauf einer Zeitdauer", wobei die Zeitdauer gleich der Entnahmezeit ist.

e) Zyklus-Modifikation

Bei der Behandlung der Wiederholungsbedingung wurde bereits angedeutet, daß es notwendig werden kann, Zyklusangaben zu modifizieren. Die Änderung kann sich auf eine Wertänderung einer Zyklusangabe beschränken - z.B. Modifikation der Zykluszeit - sie kann aber auch zu einer Änderung des Typs einer bzw. mehrerer Bedingungen führen.
Beispiel: Die Endebedingung "Uhrzeit" ist durch die Endebedingung "Unterbrechungssignal" zu ersetzen.

Eine Modifikation kann sich auf zyklische Operationen beziehen, die

- noch nicht eingeplant
- eingeplant, aber noch nicht gestartet
- eingeplant und bereits gestartet

sind.

Azyklische Erfassung

Bei dieser Art der Eingabe werden die Werte von Prozeßgrößen spontan eingelesen:

a) Die Überwachung einer Größe ist nur bei einem gewissen Prozeßzustand notwendig, eine zyklische Erfassung würde zu einer unnötigen Rechnerbelastung führen.
   Beispiel 1: Einmalige Erfassung der Mühlendrehzahl während dem kurzzeitigen Lauf der Mühle.

b) Prozeßgrößen sollen bei einem ganz bestimmten Prozeßzustand erfaßt werden, die Zeit zwischen dem Eintreten des Zustandes und dem Erfassen der Größen soll möglichst kurz sein.

   Beispiel 2: Spontane Erfassung von Prozeßgrößen zur Überwachung des ersten, kurzzeitigen Stadiums des Entnahmevorgangs nach einer Binärwertänderung der Größe EIN-TASTER

   Beispiel 3: Spontane Erfassung des Wasserkesseldrucks unmittelbar bei Einschalten der Heizung

   Beispiel 4: Nach dem Ansprechen des Grenzwertmelders muß die gestörte Prozeßgröße erfaßt werden und das Störungsprotokoll angestoßen werden.

Die azyklische Erfassung wird also bei Erfüllen einer der folgenden Bedingungen angewandt:

1. Vorliegen eines bestimmten Prozeßzustandes während des Laufs einer Task ERFASSUNG.

2. Eintreffen eines Unterbrechungssignals

   - Binärwertänderung einer Prozeßgröße (Beispiele 2 und 3)
   - Unterbrechungssignal eines Alarmgebers, beispielsweise eines Grenzwertmelders (Beispiel 4).

Entsprechend der zyklischen Erfassung kann bei Erfüllen der Bedingungen eine Verzögerung der Eingabe erforderlich sein, beispielsweise um Einschwingvorgänge oder Anlaufzeiten unberücksichtigt zu lassen.

4.6.2 Steuerung der Prozeßdatenverarbeitung

Unmittelbar nach Vorliegen eines bzw. mehrerer Analog- oder Binärwerte muß der Anstoß der Verarbeitung erfolgen. Je nach der Eingabezeit kann die Verarbeitung bereits vorliegender Werte quasiparallel zur Erfassung neuer Werte erfolgen.

4.6.3 Steuerung der Protokollierung

Anstoß der Betriebsprotokollierung

Betriebsprotokolle werden zyklisch ausgegeben. Die Zyklusstartbedingungen, Zykluszeiten und Zyklusendebedingungen entsprechen denen der Steuerung der zyklischen Prozeßdatenerfassung.

Anstoß der Störungsprotokollierung

Störungsprotokolle werden spontan, also azyklisch, nach Eintreffen einer Störung ausgedruckt und zwar nach Erfüllen einer der folgenden Bedingungen:

1. Bearbeitung von Teilaufgabe 3 (Erfassung) bzw. Teilaufgabe 4 (Verarbeitung), wobei in einem der folgenden Aufgabenelemente eine Störung festgestellt wird:

   - Analogwert- bzw. Binärwerteingabe (z.B. Störung in einer Eingabeeinheit)
   - Linearisierung mittels der Stützpunktmethode: Ein Zahlenwert liegt außerhalb der angegebenen Stützpunkte
   - Gültigkeitsprüfungen: Negative Prüfungen bei Meßkanalstörungen
   - Grenzwertprüfungen: Negative Prüfungen bei Prozeßstörungen
   - Kennwertberechnungen und Binärwertverknüpfungen: } Meßkanalstörung bei einer zu verknüpfenden Prozeßgröße.

2. Eintreffen eines Unterbrechungssignals

   - Binärwertänderung einer Binärvariablen, beispielsweise nach Ansprechen eines Überstromschutzes
   - Ansprechen eines Alarmgebers, z.B. des Grenzwertmelders zur Überwachung des Wasserkesseldrucks.

4.7 Teilaufgabe 7: Synchronisierung von Automatisierungsaufgaben

Mit der Fähigkeit eines Echtzeit-Betriebssystems, mehrere Tasks (quasi-) parallel zur Ausführung bringen zu können, ist die Notwendigkeit ihrer zeitlichen Koordinierung entstanden:
Für die Teilaufgaben eins bis fünf lassen sich drei typische Aufgabenelemente der Synchronisierung definieren.

4.7.1 Synchronisierung Prozeßdatenerfassung - Prozeßdatenverarbeitung

Bezüglich der Reihenfolge der Bearbeitung beider Teilaufgaben werden in der Literatur zwei Fälle unterschieden:

1. Einstufige Abwicklung

   Der eingegebene Wert wird unmittelbar nach dem Erfassen verarbeitet, ohne abgespeichert zu werden. Die nächste Erfassung erfolgt erst nach der vollständigen Verarbeitung des zuletzt erfaßten Wertes.

2. Zweistufige Abwicklung

   Sämtliche Werte eines Zyklus' werden nacheinander eingelesen und abgespeichert. Anschließend wird die Verarbeitung begonnen.

Diese Betrachtungsweise reicht nicht mehr aus, wenn infolge einer Verzögerungszeit bei der Adressierung bzw. der Umsetzung der Prozessor freigegeben wird. Dann nämlich wird es möglich, obige Abwicklungen zu kombinieren in eine sogenannte "parallele" Abwicklung.

3. "Parallele" Abwicklung

Die Kennzeichen lassen sich folgendermaßen darstellen:

- Anregung der Erfassung eines Wertes $x_i(kT)$ innerhalb des Zyklus', anschließend Prozessorfreigabe
- Verarbeitung bereits eingelesener Werte, höchstens bis zum Wert $x_{i-1}(kT)$, i.a. Verarbeitung nur von $x_{i-1}(kT)$ +)
- Bereitstellung und Abspeicherung des Wertes $x_i(kT)$ nach Fertigmeldung der Eingabe-Einheit.

Zeitlich gesehen läßt sich dies in der nachstehenden Abbildung verdeutlichen, wobei folgende Abkürzungen eingeführt werden (s. Bild 4.4):

(A) Aktivierung der Aufgabe VERARBEITUNG, Anstoß der Erfassung eines Wertes
(B) Fertigmeldung der Eingabe-Einheit: Abspeicherung des Eingabe-Wertes
(C) Verarbeitung bereits eingelesener Werte

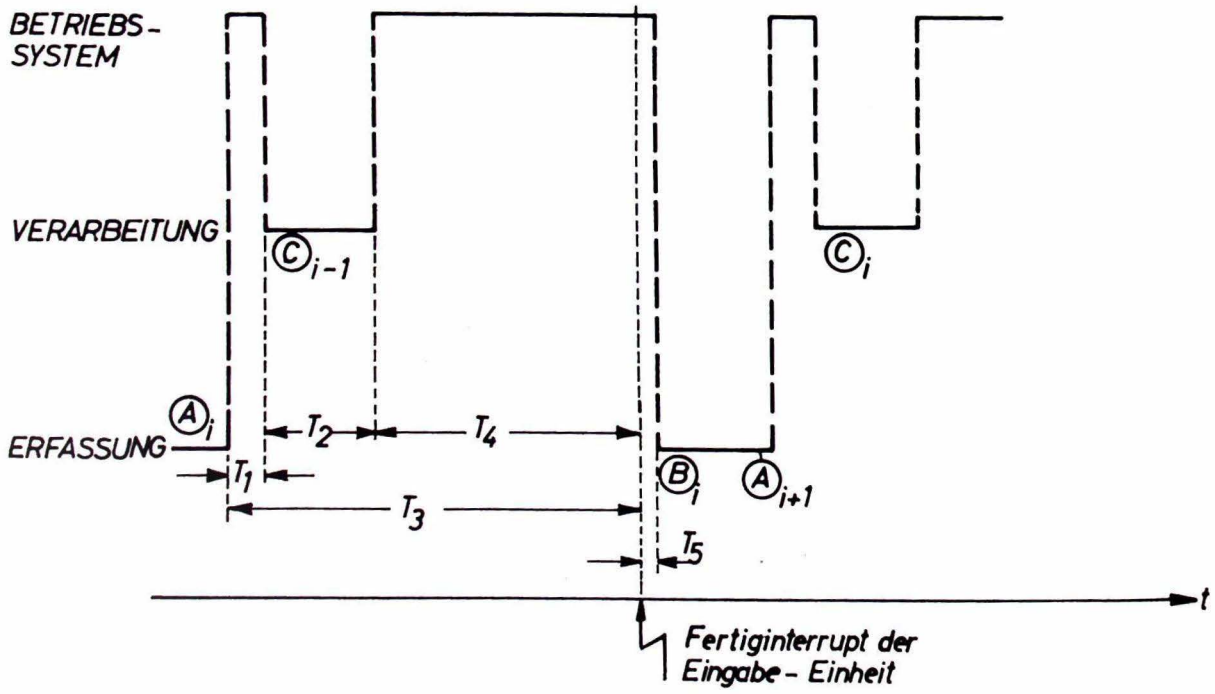


Bild 4.4: "Parallele Abwicklung"

+) Wurde die Aufgabe VERARBEITUNG aufgrund eines wichtigeren Ereignisses einmal zurückgestellt, kann es vorkommen, daß mehr als nur ein zurückliegender Wert verarbeitet werden muß.

Die in Bild 4.4 eingezeichneten Zeiten haben folgende Bedeutung:

T_1: Prozessorumschaltzeit: Task ERFASSUNG nach Task VERARBEITUNG[+)]

T_2: Zeitdauer der Verarbeitung

T_3: Eingabezeit (DIN 66 216, Blatt 4)

T_4: Prozessor-Leerlaufzeit[++)]

T_5: Prozessorvergabezeit: Prozessorleerlauf - Task ERFASSUNG

Will man keine Verzögerung der Erfassung von Werten innerhalb eines Zyklus' durch Zwischenschieben der Verarbeitung in Kauf nehmen, so ist diese Art der Bearbeitung nur unter Einhaltung der folgenden Bedingung zulässig:

$$T_3 > T_1 + T_2$$

Ob diese Bearbeitung eingesetzt wird, hängt also ab von der

- Prozessorumschaltzeit
- Zeitdauer der Verarbeitung (Laufzeit der Task VERARBEITUNG)
- Eingabezeit des Wertes.

Sehr wesentlich in diesem Zusammenhang ist die Synchronisierung dieses typischen "Erzeuger-Verbraucher"-Problems (Bild 4.5):

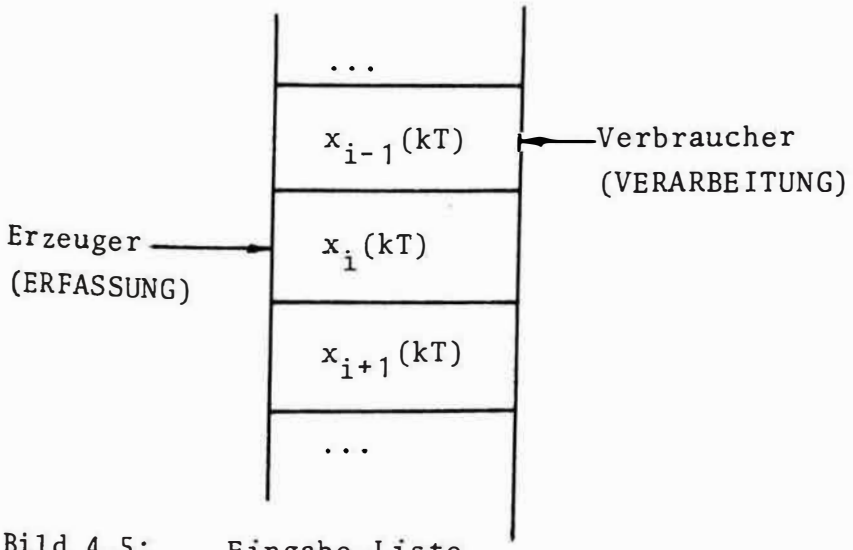


Bild 4.5: Eingabe-Liste

Um einen Überholvorgang zu vermeiden, wird gefordert, daß der Index der Verarbeitung mindestens um eins kleiner ist als derjenige der Erfassung.

+) 1. Taskzustandsänderung ERFASSUNG: "laufend" nach "E/A-unterbrochen"

2. Taskzustandsänderung VERARBEITUNG: "bereit" nach"laufend"

++) sofern keine laufbereiten Tasks vorliegen

4.7.2 Synchronisierung der Teilaufgaben Prozeßdatenverarbeitung - Datenmodellmodifikation

Ein typischer Anwendungsfall für die Synchronisierung ist die Koordinierung der Verwaltung gemeinsamer Daten, auf die also von mehreren Teilaufgaben bzw. Aufgabenelementen (quasi-) gleichzeitig zugegriffen werden kann.
Es muß sichergestellt werden, daß während der Ausführung eines Aufgabenelementes, beispielsweise der Analogwertprüfung, dort verwendete Daten - in diesem Fall die Grenzwerte - nicht geändert werden.
Im Zusammenhang mit der Teilaufgabe "Prozeßdatenverarbeitung" muß die Synchronisierung auf die folgenden Elemente angewandt werden:

1. Linearisierung von Fühlerkennlinien	-	Modifikation der Linearisierungskoeffizienten
2. Digitale Filterung	-	Modifikation des Dämpfungsfaktors
3. Zahlenanpassung	-	Modifikation der Umrechn.koeff.
4. Fertigwertkorrektur	-	Modifikation der Korrekturkoeff.
5. Grenzwertprüfung	-	Modifikation der Grenzwerte.

4.7.3 Synchronisierung der Protokollierung

Unter Synchronisierung wird hier die zeitliche Koordinierung der Protokollierung verstanden, vorausgesetzt, daß

- während der Ausgabe des Betriebsprotokolls eine bzw. mehrere Störungsmeldungen eintreffen
- während der Ausgabe des Störungsprotokolls das Betriebsprotokoll beauftragt wird
- während der Ausgabe des Störungsprotokolls weitere Störungsmeldungen eintreffen.

Im allgemeinen handelt es sich bei den Protokolliergeräten um relativ langsame E/A-Geräte, zudem ist es häufig der Fall, daß sowohl Betriebsprotokolle als auch Störungsmeldungen auf einem Gerät dokumentiert werden. Setzt man voraus, daß Störungsprotokolle sehr kurz (1..2 Zeilen), Betriebsprotokolle dagegen verhältnismäßig

lang sind, lassen sich damit folgende zeitliche Anforderungen angeben:

1. Synchronisierung der Störungsprotokollierung

Forderung 1: Keine gegenseitige Unterbrechung von Störungsprotokollen

Forderung 2: Festlegung der Reihenfolge der Störungsprotokollierung, wenn Störungen schneller eintreffen als sie dokumentiert werden können:

-1 zeitfolgerichtige Ausgabe entsprechend der zeitlichen Reihenfolge des Eintreffens

-2 Ausgabe entsprechend der Wichtigkeit der Störungen

-3 Unterdrücken noch vorgemerkter Störungsausschriebe aufgrund später eingegangener Störungsmeldungen.

2. Synchronisierung Betriebsprotokoll - Störungsprotokoll

Forderung 3: Betriebsprotokolle müssen sich von den wichtigeren Störungsprotokollen jederzeit unterbrechen lassen:

-1 Abbrechen des Betriebsprotokolls zum Zeitpunkt des Eintreffens der Störungsmeldung, nach der Ausgabe der Störungsmeldung wird die Betriebsprotokollierung neu gestartet, womit ein zusammenhängendes Protokoll erreicht wird.

-2 Unterbrechen des Betriebsprotokolls zum Zeitpunkt des Eintreffens der Störungsmeldung am Ende einer Zeile i. Das Störungsprotokoll wird eingeschoben und das Betriebsprotokoll wird bei der Zeile i+1 fortgesetzt.

Forderung 4: Störungsprotokolle dürfen sich von Betriebsprotokollen nicht unterbrechen lassen.

5. Untersuchung der PEARL-Sprachmittel zur Organisation der zyklischen Prozeßdatenerfassung

5.1 Diskussion der Sprachmittel zur Einplanung von Taskoperationen

In Kapitel 2 wurden Taskanweisungen beschrieben, bei deren Ausführung unmittelbar eine Operation wirksam wird. Soll eine Operation bei Erfüllen von Steuerbedingungen ausgeführt werden, so ist die Möglichkeit der Angabe dieser Bedingungen notwendig. Dies geschieht in PEARL dadurch, daß die Komponenten Taskoperation, Taskname und Prioritätszahl um die Komponente "Steuerbedingungen" ergänzt werden. Eine derartige Anweisung hat dann folgenden Aufbau:

Steuerbedingungen Taskoperation Taskname [Prioritätszahl] ;

Mit dieser Anweisung werden dem Betriebssystem die aufgeführten Komponenten mitgeteilt. Die auftraggebende Task wird anschließend fortgesetzt, ohne die Taskzustandsänderung des Auftragnehmers abzuwarten, die Taskoperation wird asynchron zur einplanenden Task wirksam.

Taskoperationen

Taskoperationen sind die aus Kapitel 2 bekannten Operationen, die einen Taskzustandswechsel hervorrufen.
Zusätzlich gibt es in PEARL noch eine weitere Taskoperation, die aber eine Sonderstellung einnimmt:

PREVENT [taskname]

PREVENT verursacht keine Taskzustandsänderung, sondern verhindert noch vorgemerkte bzw. zukünftige Änderungen durch das explizite Löschen aller bestehenden Einplanungen bzw. gepufferten Aufträge einer Task. PREVENT wirkt also primär nicht auf den Taskzustand und wird in diesem Sinne nicht als Taskoperation aufgefaßt.

Steuerbedingungen

Die Steuerbedingungen eines Auftragnehmers und einer zugehörigen Operation sind in PEARL in einem sog. "Schedule" zusammengefaßt. Dieser enthält sowohl die Typen als auch die Parameter der Bedingungen. Ein Schedule kann sich aus mehreren Schedule-Elementen (single schedules) zusammensetzen, die durch Kommata getrennt werden. Ein Schedule-Element wird gebildet aus Aneinanderreihung von Start-Wiederholungs-Endebedingung.

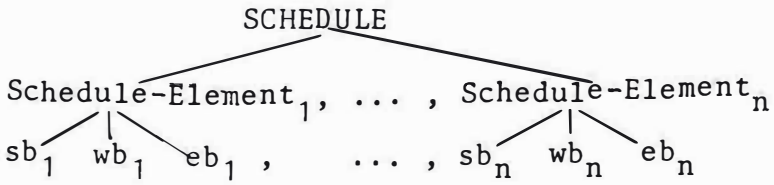
Unter Berücksichtigung der Abkürzungen

sb: Startbedingung

wb: Wiederholungsbedingung

eb: Endebedingung

ergibt sich folgendes Bild:



Das Trennzeichen ", "bedeutet disjunktive Verknüpfung von Bedingungen: Eine eingeplante Taskoperation wird immer dann ausgeführt, wenn eine Start- bzw. Wiederholungsbedingung eines Schedule-Elements erfüllt, die zugehörige Endebedingung dagegen nocht nicht erfüllt ist.
Zur Unterscheidung der Typen von Steuerbedingungen wurden in PEARL verschiedene Schlüsselworte eingeführt:

Startbedingungen:

```
AT  Uhrzeitausdruck          /* Zeitpunkt der ersten Beauftragung */
AFTER  Zeitdauerausdruck     /* Zeit bis zur ersten Beauftragung */
WHEN  Interruptausdruck      /* Zyklusstart nach Eintreffen e. Interrupts*/
```

Wiederholungsbedingung:

```
{ALL  }
{EVERY} Zeitdauerausdruck
```

Dabei wird folgender Unterschied durch die beiden Schlüsselworte ausgedrückt:

ALL : möglichst genaue Einhaltung des Zeitintervalls zwischen dem Start der Operationen

EVERY : möglichst genaue Übereinstimmung mit der absoluten Zeitskala des Rechners. Bei deren Neueinstellung kann das Zeitintervall größer oder kleiner als die angegebene Zykluszeit werden.

Endebedingungen:

```
UNTIL  Uhrzeitausdruck       /* Zeitpunkt, zu dem die letzte Beauftragung
                                spätestens noch erfolgen darf */
DURING  Zeitdauerausdruck    /* Zeit bis zur letzten Beauftragung */
```

Die mit diesen Schlüsselworten verknüpften Parameter müssen von einem der folgenden Datentypen sein:

Datentyp CLOCK

Uhrzeitdaten sind durch das Attribut CLOCK gekennzeichnet. Die Minuten- und Sekundenangaben müssen zwischen 0 und 60 liegen, die Stundenangaben werden immer modulo 24 interpretiert. Beispiel:

```
DECLARE UHRZEITSTART CLOCK;
UHRZEITSTART = 8:00:00;  /* DERSELBE WERT: 32:00:00 */
```

Die aktuelle Systemzeit kann mit der Standardprozedur NOW erfahren werden.

Datentyp DURATION

Zeitdauerdaten enthalten in PEARL das Attribut DURATION. Beispiel:

```
DECLARE VERZOEGERUNG DURATION;
VERZOEGERUNG = 8HRS 12 MIN 22.2 SEC;
```

Zuordnung von Namen zu Interrupts

Diese Zuordnung wird in PEARL im sog. SYSTEMteil vorgenommen, der die Gerätekonfiguration des Prozeßrechensystems und seine Anschlüsse an den technischen Prozeß beschreibt. Im PROBLEMteil, in dem die eigentlichen Automatisierungsprogramme enthalten sind, wird mit einer Spezifikation eine bereits bekannte Größe vereinbart.
Beispiel einer Spezifikation:

```
PROBLEM;    /* BEGINN DES PROBLEMTEILS  */
  :
SPECIFY HEIZUNGEIN INTERRUPT;
```

HEIZUNGEIN bezeichnet ein Unterbrechungssignal.

Gültigkeit eines Schedules

Grundsätzlich kann in PEARL für eine Task und eine Operation nur ein Schedule gültig sein und zwar derjenige, der zuletzt eingeplant wurde: Schedules können nicht gehäuft werden, die Verwaltung von Schedules durch das Betriebssystem erfolgt task- und operationsspezifisch. Eine bestehende Einplanung wird bei einer erneuten Einplanung derselben Operation für dieselbe Task implizit gelöscht. Beispiel:

```
EINPLANUNG1: ALL 5 SEC UNTIL 18:00:00 ACTIVATE ERFASSUNG;
EINPLANUNG2: WHEN INTERRUPT ALL 1 SEC ACTIVATE ERFASSUNG;
EINPLANUNG3: AFTER 1 SEC ALL 5 SEC ACTIVATE DIGITAL;
```

Bei Ausführung der zweiten Einplanung wird die erste gelöscht, die zweite Einplanung dagegen wird nach Ausführung der dritten infolge eines anderen Auftragnehmers nicht gelöscht.
Durch die ausschließliche Verwendung der Operation ACTIVATE wirkt sich bei der Prozeßdatenerfassung nur die taskspezifische Verwaltung von Schedules aus, für eine Task ERFASSUNG kann nur ein Schedule existieren. Konsequenzen dieser Einplanungsvorschrift betreffen die Modifikation von Schedules sowie das Unterbrechen, Fortsetzen und Beenden der zyklischen Auftragserteilung. Sie werden in den entsprechenden Abschnitten diskutiert.

Zustände eines Schedules

Zustand bedeutet hier Grad der Bekanntheit der einzelnen Steuerbedingungen durch das Betriebssystem. Die Folge davon ist, daß abhängig vom Zustand das Erfüllen einer Start- bzw. Wiederholungsbedingung eine Operation auslöst oder das Erfüllen einer Endebedingung zur Einstellung eines zyklischen Auftrags führt.

Ein Schedule +) einer noch nicht eingeplanten Operation hat den Zustand "unbekannt". Das Erfüllen einer Bedingung veranlaßt keine Aktion. Beispiel:

```
/* Das Eintreffen des Interrupts "HEIZUNGEIN" vor der nachfolgenden
   Einplanung führt nicht zum Zyklusstart */
:
WHEN HEIZUNGEIN ALL 5 SEC ACTIVATE ERFASSUNG;
```

Ein Schedule erfährt durch eine Einplanung eine Zustandsänderung von "unbekannt" nach "bekannt".
Der Zustand "bekannt" bedeutet für eine

Startbedingung: Das Erfüllen der Startbedingung führt erstmals zu einem Auftrag.

Wiederholungsbedingung: Jedes Erfüllen der Wiederholungsbedingung führt zu einem Auftrag.

Endebedingung: Die Endebedingung ist noch erfüllbar; das Erfüllen führt zu keinem Auftrag sondern zur Einstellung der Auftragserteilung.

Ist eine Startbedingung "bekannt", so muß die zugehörige Wiederholungs- bzw. Endebedingung zwingendermaßen ebenfalls "bekannt" sein.

Ist ein Schedule "bekannt", so kann er durch implizites oder explizites Löschen der Einplanung wieder "unbekannt" gemacht werden.
Beispiel:

```
AT 8:00:00 ALL 2 SEC ACTIVATE ERFASSUNG; /* ZUSTAND "BEKANNT" */
:
PREVENT ERFASSUNG;  /* ZUSTAND "UNBEKANNT" */
```

Neben den Zustandsänderungen durch das Ausführen von Anweisungen kann das Erfüllen von Steuerbedingungen zu einer Zustandsänderung führen:

+) einschließlich all seiner Schedule-Elemente und Steuerbedingungen

Zustandsänderung eines Schedules bei Erfüllen einer Startbedingung

1. Durch das Erfüllen der Startbedingung der Typen "Leerbedingung"[+], "Erreichen einer Uhrzeit" bzw. "Ablauf einer Zeitdauer" wird die Startbedingung "unbekannt".
   Für die Startbedingung "Erreichen einer Uhrzeit" ist die Konsequenz diejenige, daß nach der Abarbeitung der Startbedingung die Operation - bei ausdrücklichem Wunsch - für den folgenden Tag neu eingeplant werden muß.
   Beispiel:
   AT 8:00:00 ALL 2 SEC ACTIVATE ERFASSUNG;

   Nach 8 h ist die Startbedingung "abgearbeitet", d.h. dem Betriebssystem "unbekannt" und demgemäß am nächsten Tag unwirksam, der Schedule ist nur noch"teilweise bekannt":
   Ist zumindest eine Steuerbedingung eines Schedules "unbekannt" und noch wenigstens eine Steuerbedingung dieses Schedules "bekannt", wird dieser als "teilweise bekannt" bezeichnet.

2. Startbedingungen des Typs "Eintreffen eines Interrupts" - evtl. mit einer Verzögerungszeit kombiniert - bleiben dagegen auch nach ihrem Erfüllen "bekannt". Als Folge davon bleiben Wiederholungs- und Endebedingung des zugehörigen Elements ebenfalls "bekannt".
   Die Startbedingung sowie das zugehörige Schedule-Element ist so lange "bekannt", bis die Einplanung implizit bzw. explizit gelöscht wird. Beispiel:
   SPECIFY HEIZUNGEIN INTERRUPT;
   WHEN HEIZUNGEIN ALL 5 SEC DURING 2 MIN ACTIVATE ERFASSUNG;

   Sowohl nach Eintreffen von HEIZUNGEIN als auch nach Ablauf der Zeitdauer von zwei Minuten bleibt die Startbedingung "bekannt", ein erneuter Interrrupt führt zu einem erneuten Zyklusstart.

Zustandsänderung eines Schedules bei Erfüllen einer Endebedingung

Bei Erreichen eines Endezeitpunkts bzw. dem Ablauf einer Zeitdauer (Schlüsselworte UNTIL bzw. DURING) sind im Zusammenhang mit den unterschiedlichen Startbedingungen zwei Fälle möglich:

1. Startbedingung "Eintreffen eines Interrupts"

   Auch nach dem Zyklusende bleibt der Schedule "bekannt" (s. obiges Beispiel).

+) Wird eine Startbedingung nicht explizit angegeben ("Leerbedingung"), so wird der Zyklus unmittelbar nach der Einplanung gestartet

2. Andere Startbedingungen

Nach Erreichen der Endebedingung wird der Schedule (bzw. das entsprechende Schedule-Element) "unbekannt".
Beispiel:
ALL 2 MIN UNTIL 18:00:00 ACTIVATE ERFASSUNG;
Nach 18 h wird der Schedule "unbekannt".

Zustandsdiagramm eines Schedules

Aus den Aussagen über die Operationseinplanung, Startbedingungen und Endebedingungen ergibt sich ein Schedule-Zustandsdiagramm für eine Operation und einen Auftragnehmer entsprechend Bild 5.1.
Aus Gründen der Übersichtlichkeit wurde dabei angenommen, daß der Schedule aus einem Schedule-Element besteht.
Die Abkürzungen haben folgende Bedeutung:

u : "unbekannt"
tb: "teilweise bekannt"
b : "bekannt"

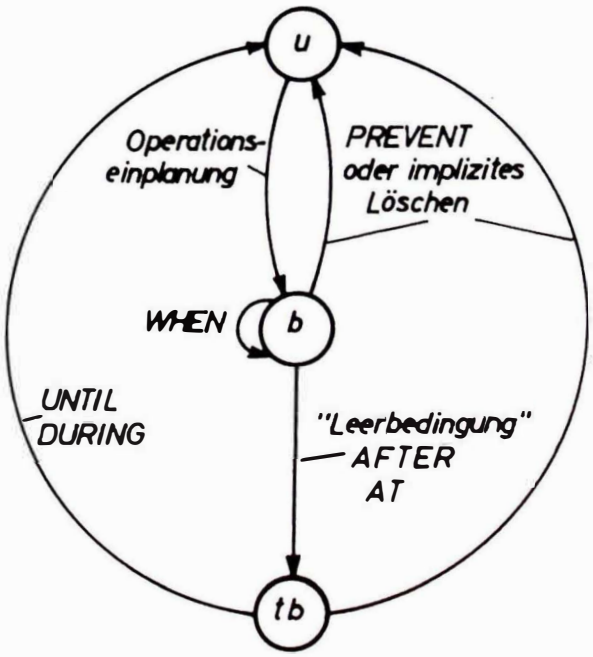


Bild 5.1: Schedule-Zustandsdiagramm

Erläuterungen zu Bild 5.1:

Durch eine Operationseinplanung wird ein Schedule "bekannt". Er bleibt auch bei Erfüllen einer evtl. vorhandenen Startbedingung des Typs "Interrupt" in diesem Zustand - im Gegensatz zu den Startbedingungen der Typen "Leerbedingung", "Ablauf einer Zeitdauer" und "Erreichen einer Uhrzeit": In diesen Fällen wird ein Übergang nach "teilweise bekannt" erreicht.
Nach Erfüllen einer evtl. angegebenen Endebedingung wird der Schedule in den Zustand "unbekannt" überführt, ebenso wie durch implizites oder explizites Löschen der Einplanung.

Die Zustandsübergänge werden also vorgenommen bei

- Ausführung von Anweisungen (Einplanung, PREVENT, implizites Löschen)
- Erfüllen von Steuerbedingungen (AT, AFTER, DURING, UNTIL).

Zustände eines Zyklus

Ähnlich den Taskzuständen lassen sich zur Verwaltung zyklischer Operationen sogenannte "Zykluszustände" definieren. Bei der Programmierung des Modellprozesses, insbesondere bei der Behandlung der Modifikation der Typen bzw. Parameter von Steuerbedingungen der Schedules [32] , [49] hat sich die Notwendigkeit einer Einführung derartiger Zustände ergeben.
Man unterscheidet zwischen zwei Zuständen:

1. "passiv"
   Die Startbedingung eines Schedule-Elements einer eingeplanten Operation ist noch nicht erfüllt worden bzw. die zyklische Auftragserteilung ist durch das Erfüllen der zugehörigen Endebedingung oder dem Löschen der Einplanung beendet worden.
2. "aktiv"
   Voraussetzungen für diesen Zustand sind die folgenden:
   - Einplanung einer Operation
   - Erfüllung der Startbedingung
   - Nicht-Erfüllung der zugehörigen Endebedingung.

Änderung des Zykluszustands durch die Auftragserteilung

Bei Erfüllen von Start- bzw. Wiederholungsbedingungen "bekannter" Schedule-Elemente können unterschiedliche Operationen auf eine Task zyklisch angewandt werden:

Zyklen werden task-, operations- und scheduleelementspezifisch verwaltet.

Dies soll anhand eines Beispiels verdeutlicht werden:

sb_{11} wb_{11} eb_{11}, sb_{12} wb_{12} eb_{12}, sb_{13} wb_{13} eb_{13} ACTIVATE TA;
sb_{21} wb_{21} eb_{21} TERMINATE TA;

Nach der Ausführung einer Einplanung, also vor Erfüllen der Startbedingungen, werden die Zustände der Zyklen bezüglich einer Task, einer Operation und der zugehörigen Schedule-Elemente als "passiv" bezeichnet. Am Beispiel bedeutet das: Sowohl die Zustände der Zyklen für die Task TA, der zugehörigen Operation ACTIVATE und der zugehörigen Schedule-Elemente 1 und 2 sind "passiv" wie auch der Zyklus für die Task TA, der Operation TERMINATE und des Elements 1.
Wird nun eine Startbedingung eines Schedule-Elements, beispielsweise sb_{11} erfüllt, wird die entsprechende Operation - hier ACTIVATE - zyklisch auf die Task TA angewandt. Der Zustand des Zyklus' bezüglich der Task TA, der Operation ACTIVATE und des Schedule-Elements 1 wird als "aktiv" bezeichnet und zwar so lange, bis die zugehörige Endebedingung - hier eb_{11} - eintrifft bzw. die Einplanung gelöscht wird.

Bei der Prozeßdatenerfassung entfällt durch die ausschließliche Verwendung von ACTIVATE die operationsspezifische Verwaltung von Zyklen, es läßt sich dann unter Anwendung der obigen Einplanung zum Beispiel folgendes Zyklusdiagramm bezüglich der Task TA und der Operation ACTIVATE angeben (s. Bild 5.2):

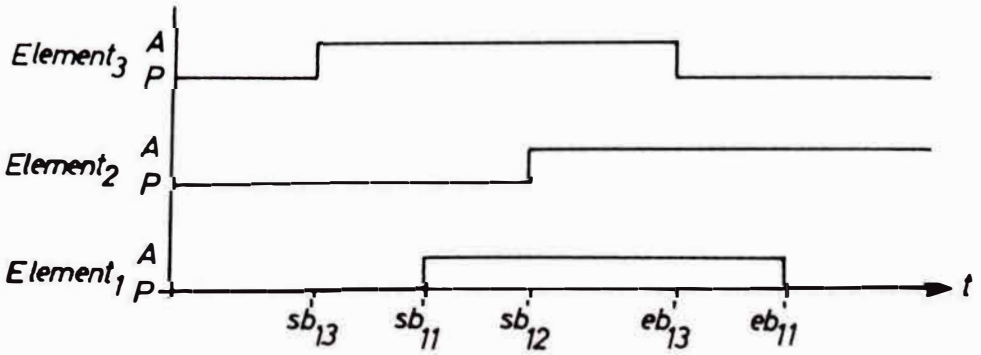


Bild 5.2: Zyklusdiagramm der Task TA und der Operation ACTIVATE (A = "Aktiv", P = "Passiv")

Eine ausschließlich zyklische Auftragserteilung erfolgt nur zwischen $sb_{13} < t < sb_{11}$ und bei $t > eb_{11}$.

5.2 Untersuchung und Beurteilung der Sprachmittel zur Einplanung der zyklischen Auftragserteilung

Im letzten Abschnitt wurde gezeigt, welche Sprachmittel in PEARL für diese Aufgabe zur Verfügung stehen. Sie werden im folgenden bei ihrer Anwendung auf den Modellprozeß gemäß den Anforderungen untersucht und beurteilt. Dazu wird in einem ersten Punkt auf die Einplanung selbst eingegangen, während die Sprachelemente für die zahlreichen Steuerbedingungen in den nachfolgenden Abschnitten diskutiert werden.

Realisierung der Einplanung

[Schedule] { ACTIVATE | TERMINATE | SUSPEND | CONTINUE | RESUME | PREVENT } Taskname [Prioritätszahl] ;

Implementierung

Die Implementierungen unterscheiden sich dahingehend, daß gewisse Steuerbedingungen nur mit bestimmten Taskoperationen verknüpft werden dürfen bzw. daß gewisse Operationen mit keinem Schedule versehen werden können.
Beispiel:
Im ASME-PEARL-Subset Stufe 1 (APS1) und in Basis-PEARL [43] sind folgende Einschränkungen vorhanden:

- Zyklische Aktionen können ausschließlich mit ACTIVATE eingeplant werden
- CONTINUE und RESUME sind nur mit Startbedingungen[+)] zulässig
- TERMINATE, SUSPEND und PREVENT dürfen nicht mit einem Schedule formuliert werden.

Sprachuntersuchung

1. Einplanungszeitpunkt

   Die Zuordnung von Steuerbedingungen zu Auftragnehmer und Taskoperation wird dem Betriebssystem zur Ausführungszeit der Anweisung ("dynamisch") übergeben und kann je nach Anforderung durch eine erneute Einplanung bei einem genau bestimmbaren Prozeßzustand - also gezielt - ungültig gemacht bzw. modifiziert werden.

+) Sie bedeuten im Zusammenhang mit CONTINUE/RESUME Fortsetzungsbedingungen

Beispiel:

```
DECLARE ENTNAHMEPHASE BIT(1);
.
.
IF ENTNAHMEPHASE THEN ALL ZYKLUS/2 ACTIVATE ERFASSUNG;
                FIN;
```

Im Betriebszustand "Entnahmephase" wird die Zykluszeit halbiert.

Diese Vorgehensweise steht im Gegensatz zur "statischen" Zuordnung von Steuerbedingungen zu Auftragnehmer, wie beispielsweise die Interruptzuordnung bei Assemblerprogrammen der AEG 60-50:
Die Zuordnung erfolgt dort per Korrespondenz über die Bedienungsschreibmaschine, kann also von einem Programm aus nicht gezielt geändert werden (statische Zuordnung).
Als wesentlicher, bei der Einplanung in PEARL besonders zu berücksichtigender Gesichtspunkt stellt sich damit die folgende Tatsache heraus:
Mit einer Taskinganweisung werden die zum Zeitpunkt der Einplanung gültigen Typen und Parameter des Schedules dem Betriebssystem übermittelt. Spätere Änderungen des Schedules ohne erneute Einplanung haben auf die bereits eingeplante Operation keinen Einfluß.
Die Nichtberücksichtigung dieses Gesichtspunkts führte insbesondere bei der Zyklusmodifikation zu Programmierfehlern (s. 5.7).
Beispiel:

```
DECLARE ZYKLUS DURATION INITIAL (4 SEC);
.
.
EINPLANUNG: ALL ZYKLUS ACTIVATE ERFASSUNG;
.
.
AENDERUNG: ZYKLUS = ZYKLUS/2;
```

Je nach Reihenfolge der Ausführung der beiden Anweisungen erfolgt eine Zykluszeitänderung.

Reihenfolge:	1. "Einplanung" 2. "Änderung"	} keine Zykluszeitänderung
Reihenfolge:	1. "Änderung" 2. "Einplanung"	} Zykluszeitänderung

2. Taskoperationen

Bei der Programmierung der Einplanung der zyklischen Erfassung hat sich gezeigt, daß sich die Anforderung dieser Aufgabe auf das Einleiten einer Tätigkeit beschränkt und damit durch die Operation ACTIVATE voll erfüllt wird. Die übrigen, in Kapitel 2 aufgeführten Taskoperationen, wurden im Zusammenhang mit Wiederholungsbedingungen nicht benötigt. Die Beschränkung im APS1, Wiederholungsbedingungen ausschließlich mit ACTIVATE anwenden zu können, hat sich u.a. an dieser Stelle nicht als einschränkend erwiesen und wurde auch in Basis-PEARL übernommen.

3. Auftragnehmer

In PEARL kann je Auftrag eine Operation bei nur einem Auftragnehmer eingeplant werden.
Eine denkbare Aufgabe, daß eine Operation bei gleichen Start-, Wiederholungs- und Endebedingungen zum gleichen Zeitpunkt für zwei oder mehr Tasks eingeplant werden soll, muß wohl als Sonderfall angesehen werden. Bei der Programmierung der Teilaufgaben des Modellprozesses, u.a. [32], [39] und [50], zeigte sich hierfür keine Anwendung. Diese Aufgabe kann in PEARL durch Aneinanderreihung von Einplanungen mit verschiedenen Auftragnehmern gelöst werden. Nachteilig ist die mögliche Unterbrechbarkeit der auftraggebenden Task zwischen den Anweisungen.

4. Erfolgreiche und erfolglose Einplanung

Bei der Einplanung der zyklischen Prozeßdatenerfassung traten während des Ablaufs von Automatisierungsprogrammen des Modellprozesses Fehler auf, die auch zu einer Implementationsänderung der Zeitverwaltung führten, wenn die Zeitparameter der Steuerbedingungen prozeßzustandsabhängige Variable waren [49] .

Demzufolge lassen sich bezüglich des Erfolges oder Mißerfolges einer Einplanung zwei Fälle unterscheiden:

a) Einplanungs-Erfolg

Im Gegensatz zu Taskinganweisungen ohne Bedingungsangabe kann eine Einplanung mit korrekten Parametern der Bedingungen unabhängig vom Taskzustand des Auftragnehmers immer stattfinden. Korrekt werden Parameter bezeichnet, wenn die Werte, die sich aus der Berechnung der für die Bedingungen verwendeten Ausdrücke ergeben, plausibel sind.

Beispiel:

```
DECLARE ZYKLUS DURATION INITIAL (2 SEC);
.
.
FAKTOR = 2;
ALL (FAKTOR ZYKLUS) ACTIVATE ERFASSUNG; /* EINPLANUNG */
.
.
FAKTOR = FAKTOR-3;  /* FAKTOR : -1 : */
ALL (FAKTOR  ZYKLUS) ACTIVATE ERFASSUNG;
/* DIE ZYKLUSZEIT IST NICHT PLAUSIBEL */
```

b) Einplanungs-Mißerfolg

In PEARL wird keine Aussage[+)] gemacht über die Behandlung einer Einplanung bei nicht plausiblen Parametern; möglich und notwendig ist eine Negativquittung einer fehlerhaften Einplanung durch ein SIGNAL[++)], damit der Benutzer den Einplanungs-Mißerfolg erfahren kann (s. Beispiele in 5.4 ff).

Folgende Varianten und deren Lösungswege sind im Fehlerfall zu unterscheiden:

- Ein Schedule besteht aus einem Schedule-Element (APS1):
  Ist mindestens ein nicht plausibler Parameter vorhanden, so ist keine Einplanung möglich (s. obiges Beispiel)
- N Schedule-Elemente vorhanden (N > 1):
  Fall 1: Nicht plausible Parameter in M Elementen (M<N)
  1. Alternative:
  Die Operation wird unter Berücksichtigung der (N-M) korrekten Schedule-Elemente eingeplant. Wertung: Unsicherheit, welche Elemente berücksichtigt werden und welche nicht.

+) Die hier aufgeführte Forderung nach einer Negativquittung bei nicht plausiblen Parametern ist auf Anregung des Verfassers nachträglich in die Sprachbeschreibung aufgenommen worden ([42] Sektion 2.4/25).

++) Die Reaktion auf ein SIGNAL ist frei programmierbar; wenn keine Reaktion programmiert ist, tritt eine implementationsabhängige Reaktion ("Systemreaktion") in Kraft.

2. Alternative:
Die Operation wird nicht eingeplant. Wertung:
Diese eindeutige Reaktion ist der ersten Alternative vorzuziehen[+++].

Fall 2: Nicht plausible Parameter in jedem der N Elemente:
In diesem Fall ist grundsätzlich keine Einplanung möglich.

Beurteilung

PEARL enthält ein Sprachelement, das für ein gezieltes Einplanen der zyklischen Prozeßdatenerfassung gut geeignet ist. Die ausschließliche Verwendung der Operation ACTIVATE im Zusammenhang mit Wiederholungsbedingungen sowie die Einplanung eines Auftragnehmers je Auftrag hat sich als ausreichend erwiesen. Vorteilhaft ist, daß dem Anwender bei nicht plausiblen Parametern der Steuerbedingungen bereits bei der Einplanung eine Mißerfolgs-Rückmeldung geliefert wird.

5.3 Untersuchung und Beurteilung der Sprachelemente für die Wiederholungsbedingung

In diesem sowie den folgenden Abschnitten werden die Steuerbedingungen im Zusammenhang mit zyklischer Auftragserteilung diskutiert, d.h. ein Schedule besitzt in jedem seiner Elemente eine Wiederholungsbedingung. Als Kennzeichen einer zyklischen Beauftragung soll sie an erster Stelle genannt werden.

Realisierung

{ ALL / EVERY } Zeitdauerausdruck

Implementierung

Die Implementierungen können sich insbesondere bezüglich des Wertebereiches der Zykluszeiten unterscheiden:
In PEARL sind Angaben von Stunden, Minuten und Sekunden einschließlich evtl. Sekundenbruchteilen möglich, über kleinste und größte Werte wird keine Aussage gemacht. Die kleinste Zeitdauer bei der AEG 60-50 Implementierung beträgt durch die Verwendung vorhandener Zeitverwaltungsfunktionen eine Sekunde, die größte Zeitdauer war durch die Rechnerwortlänge mit 8 388 607 Sekunden gegeben, sie wurde aber infolge der abendlichen Rechnerabschaltung auf 18 Stunden begrenzt.

+++) Diese Lösung wurde in die Sprachbeschreibung aufgenommen.

Eine weitere Implementationsabhängigkeit kann durch eine Beschränkung des Zeitdauerausdrucks vorliegen bzw. eine Beschränkung des Datentyps DURATION auf einfache Datenelemente (APS1, s. nachstehende Beispiele).

Sprachuntersuchung

1. Zykluszeit

Die kleinste Zykluszeit von 1 Sekunde hat sich bei der überwiegenden Zahl der Überwachungsaufgaben am Modellprozeß als ausreichend erwiesen. Eine Ausnahme bildete die Überwachung einiger Binärgrößen während der Entnahmephase, in der Zeiten von 0,2 ... 0,5 Sekunden für die Zyklusstartverzögerung bzw. Zykluszeit zu einer zuverlässigen Überwachung notwendig gewesen wären [49].

In PEARL ist die Angabe eines Zeitdauerausdrucks für variable, prozeßzustandsabhängige Zykluszeiten möglich. Die Anwendungen, u.a. beim nachstehenden Beispiel des Wasserkesseldrucks, haben gezeigt, daß es in diesem Fall vor der Einplanung unbedingt erforderlich ist, den Ergebniswert des Zeitdauerausdrucks auf einen entsprechenden Wertebereich hin zu überprüfen. Bei nicht plausiblen Parametern kann keine Einplanung erfolgen (Fehlerrückmeldung durch ein SIGNAL bei versuchter Einplanung).

Beispiel:

Je nach dem, ob die Wasserkesselheizung ein- oder ausgeschaltet ist, muß die Zykluszeit verkleinert werden:

```
DECLARE ZYKLUS DURATION, FAKTOR FIXED, HEIZUNGSZUSTAND BIT(1);
                                       /* "1": HEIZUNG EIN */
                                       /* "0": HEIZUNG AUS */
FAKTOR = 3;
.
.
IF HEIZUNGSZUSTAND THEN
   IF ZYKLUS < (FAKTOR * 1 SEC) OR ZYKLUS > (FAKTOR * 18 HRS)
   THEN GOTO ALARM;
   ELSE ALL ZYKLUS/FAKTOR ACTIVATE ERFASSUNG;
   FIN;
FIN;
```

- Konstante Zykluszeit

Unter Verwendung des Zugriffsattributs INV (invariant) wird nur ein lesender Zugriff auf Daten erlaubt. Die Initialisierung muß bei der Vereinbarung erfolgen.

Beispiel:

```
DECLARE TIME INV DURATION INITIAL (2 SEC);
ALL TIME ACTIVATE ERFASSUNG;
```

Mit einer konstanten Zykluszeit wurde der größte Teil der Binärgrößen des Modellprozesses erfaßt, ebenso wie einige langsam veränderliche Analoggrößen, wie beispielsweise Dampfkesseldruck, Dampfkesselfüllstand usw.

- Variable Zykluszeit

  Durch das Zulassen eines Zeitdauerausdrucks ist es möglich, die Zykluszeiten entsprechend dem Prozeßzustand zu berechnen (s. obiges Beispiel und Abschnitt 5.7).

2. Beschränkung des Datentyps DURATION

Die im APS1 vorhandene Beschränkung des Datentyps DURATION auf einfache Datenelemente hat sich bei der Programmierung des Modellprozesses als einschränkend und unzweckmäßig herausgestellt (s. Beispiel und [49]) und wurde in Basis-PEARL nicht übernommen.
Beispiel:
Vereinbarung und Initialisierung von Daten für die meßstellenspezifischen Zykluszeiten für Analoggrößen.

```
DECLARE ANALOGZYKLUS(3) DURATION INIT (2 SEC, 4 SEC, 4 SEC);
```

Bei Fehlen von DURATION-Feldern sind zwei Realisierungsmöglichkeiten zu unterscheiden:

- Beschränkung auf einfache Datenelemente mit unterschiedlichen Bezeichnern; nachteilig ist hierbei die umständliche Programmierung.
- DURATION -Felder werden durch Felder des Datentyps FIXED bzw. FLOAT ersetzt; diese Daten müssen vor ihrer Verwendung jedesmal in Zeitdauerdaten umgerechnet werden. Beispiel:

```
DECLARE ZEIT DURATION;
DECLARE ANALOGZYKLUS(3) FIXED INITIAL (2,4,4);
.
.
ZEIT = ANALOGZYKLUS(I)* 1 SEC;
```

3. Entkopplung von Einplanung und Prozeßdatenerfassung

Durch das Vorhandensein eines Sprachelements für die Wiederholungsbedingung ist es möglich, die Formulierung der zeitlichen Steuerung und der eigentlichen Erfassung zu entkoppeln. Eine Task ERFASSUNG braucht keine Sprachelemente zu ihrem Anstoß enthalten, kann also sowohl für die zyklische als auch azyklische Prozeßdatenerfassung verwendet werden (s. Bild 5.3).

Im Gegensatz hierzu steht die Kopplung von Zeitsteuerung und Erfassung, wie beispielsweise bei Assemblerprogrammen unter dem Betriebssystem MARVIS: Es gibt dort keine Möglichkeit einer direkten Zyklussteuerung, diese muß bei jedem Programmlauf auf einen einmaligen, um die Zykluszeit verzögerten Programmstart abgebildet werden, was natürlich einen zusätzlichen Organisationsaufwand in der Zeitverwaltung mit sich bringt.
Die Zeitsteuerung und Erfassung ist in diesem Fall gekoppelt, d.h. man legt sich bei der Programmierung der Task ERFASSUNG auf die zeitliche Steuerung fest (s. Bild 5.4). Will man hier sowohl eine azyklische als auch zyklische Erfassung erreichen, so muß im ersteren Fall die wiederholte Einplanung durch das Setzen von vorgesehenen Merkern verhindert werden.

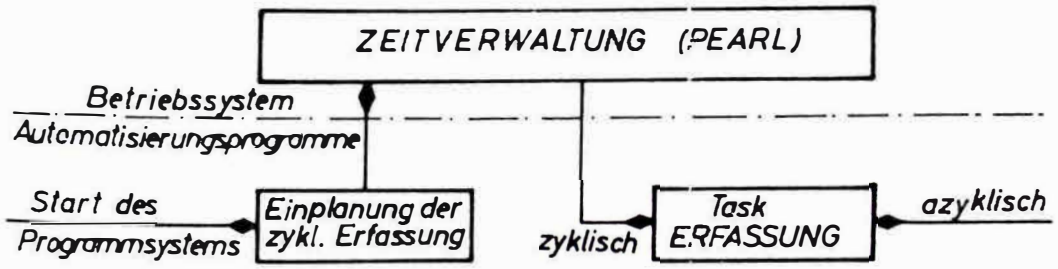


Bild 5.3: Entkopplung von Zeitsteuerung und Erfassung

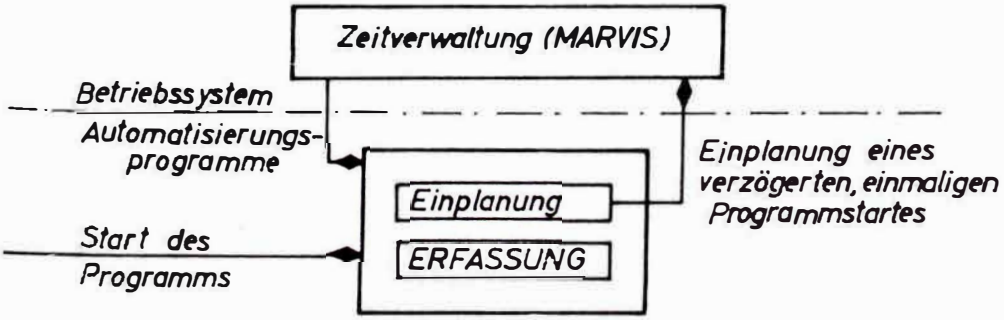


Bild 5.4: Kopplung von Zeitsteuerung und Erfassung

Beurteilung

Die gestellten Anforderungen werden durch die Schlüsselworte ALL bzw. EVERY sowie dem Datentyp DURATION gut erfüllt. Die Beschränkung dieses Datentyps auf einfache Datenelemente hat sich als nachteilig erwiesen. Sowohl konstante als auch variable Zykluszeiten sind möglich. Nicht realisierbare Zykluszeiten werden dem Benutzer bei versuchter Einplanung rückgemeldet.

5.4 Untersuchung und Beurteilung der Sprachelemente für Zyklus-Startbedingungen

5.4.1 Sofortiger Zyklusstart nach der Einplanung

Realisierung

Die zeitliche Kopplung von Zyklusstart und Zykluseinplanung wird in PEARL durch Weglassen einer explizit formulierten Startbedingung erreicht ("Leerbedingung"). Weglassen einer Startbedingung impliziert also sofortigen Zyklusbeginn. Das zugehörige Schedule-Element beginnt demzufolge mit dem Schlüsselwort ALL bzw. EVERY. Der Zyklusstart ist ausschließlich vom Einplanungszeitpunkt abhängig.

Implementierung

Diese spezielle Startbedingung braucht nicht gesondert implementiert zu werden. Sie ist auf einen zeitverzögerten Zyklusbeginn zurückzuführen, wobei in diesem Fall die Zeitverzögerung 0 Sekunden beträgt (s. 5.4.2 und [8]).

Sprachuntersuchung

Als typischer Anwendungsfall ergab sich beim Modellprozeß die Einplanung einer Operation zur zyklischen Prozeßdatenerfassung im Anschluß an die Eingabe der meßstellenspezifischen Daten über den Kartenleser (Datenmodell-Initialisierung).
Voraussetzung der Anwendung dieser Startbedingung ist, daß überprüft wird, ob eine zyklische Erfassung bzw. Überwachung zum Zeitpunkt der Einplanung bereits erfolgen darf bzw. sinnvoll ist. Am obigen Beispiel bedeutet dies: Vor der Einplanung muß überprüft werden, ob sich das Modell bereits in dem zu überwachenden Zustand befindet, beispielsweise in der Arbeitsphase. Als vorteilhaft hat sich erwiesen, daß aus dem Weglassen der Formulierung der Startbedingung der sofortige Zyklusbeginn unmittelbar ersichtlich ist.

Beurteilung

PEARL wird der Forderung, bei sofortigem Zyklusbeginn die Startbedingung nicht explizit formulieren zu müssen, gerecht.

5.4.2 Zeitverzögerter Zyklusstart

Realisierung

Die Einplanung eines zeitverzögerten Zyklus' kann mit der Steuerbedingung 'AFTER Zeitdauerausdruck' ermöglicht werden.
Der Zeitpunkt des Zyklusbeginns wird durch Addition des Zeitpunkts der Einplanung und der angegebenen Zeitdauer festgelegt, die Erfassung beginnt also um eine relative Zeitdauer bezüglich des Einplanungszeitpunkts verspätet.

Implementierung

Bei den Implementierungen sind zwei Fälle zu unterscheiden:

1. Nächtliche Rechnerabschaltung

   Sinnvoller Wertebereich des Zeitdauerausdruck-Ergebniswertes:

   $0 \leqq$ Zeitdauer $\leqq 18$ Stunden

   Eine Verletzung dieses Wertebereichs muß bei versuchter Einplanung ein SIGNAL zur Folge haben.

2. Durchgehender Rechnerbetrieb

   In diesem Fall ist eine Überprüfung der Zeitdauer auf einen Wert von $\geqq 0$ Sekunden ausreichend. Bei negativer Zeitdauer hat bei versuchter Einplanung ebenfalls ein SIGNAL zu erfolgen.

Sprachuntersuchung

Die Anwendungen haben gezeigt, daß bei variabler, prozeßzustandsabhängiger Zeitverzögerung eine Überprüfung des Zeitdauerwertes auf die erwähnten Wertebereiche erforderlich ist, will man immer eine erfolgreiche Einplanung erreichen.
Beispiel:
Start der zyklischen Überwachung der Arbeitsphase des Modellprozesses nach Beenden der Anlaufphase (Zeitdauer ANLAUFZEIT):

```
DECLARE ANLAUFZEIT DURATION;
.
.
IF ANLAUFZEIT < 0 SEC OR ANLAUFZEIT > 18 HRS THEN GOTO ALARM;
ELSE AFTER ANLAUFZEIT ALL 2 SEC ACTIVATE ERFASSUNG; FIN;
```

Beurteilung

PEARL läßt durch die Bedingung 'AFTER Zeitdauerausdruck' konstante und variable Zyklusverzögerungen zu, erfüllt also die Anforderungen. Eine nicht realisierbare Startbedingung wird dem Benutzer durch ein SIGNAL rückgemeldet.

5.4.3 Zyklusstart nach Erreichen einer Uhrzeit

Im Zusammenhang mit dem Parameter "Uhrzeitausdruck" wurde zur Realisierung dieser Startbedingung das Schlüsselwort AT eingeführt.

Implementierung

1. Bezüglich der Genauigkeit von Uhrzeiten werden in PEARL zumindest Stunden-, Minuten- und Sekundenangaben gefordert und bei der AEG 60-50 Implementierung auch derart realisiert.
   Zur Erhöhung der Genauigkeit ist eine Unterteilung der Sekundenangaben möglich.

2. Nächtliche Rechnerabschaltung

   Eine Einplanung für den nächsten Tag ist nicht möglich. Das Ergebnis des Uhrzeitausdrucks muß daher auf einen entsprechenden Wertebereich hin überprüft werden. Bei versuchter Einplanung muß im Falle einer Verletzung des Wertebereichs ein SIGNAL erzeugt werden.

3. Durchgehender Rechnerbetrieb

   Eine Einplanung für denselben und den nächsten Tag ist möglich - abhängig vom Ergebniswert T_1 des Uhrzeitausdrucks und der aktuellen Systemzeit NOW:
   NOW $\leqq T_1$: Zyklusstart am selben Tag
   NOW $\geqq T_1$: Zyklusstart am nächsten Tag.

   Eine Überprüfung des Uhrzeitwertes von der Zeitverwaltung ist nicht möglich und nicht notwendig, die Einplanung - restliche Parameter des Schedules als korrekt vorausgesetzt - ist immer erfolgreich.

Sprachuntersuchung

Durch die nächtliche Rechnerabschaltung traten bei der Anwendung auf die Modellprozeßüberwachung die beiden Fälle auf:

- Erfolgreiche Einplanung für denselben Tag
- Einplanungs-Mißerfolg.

Aus letzterem Grund wurde ein variabler Parameter dieser Startbedingung immer auf Plausibilität überprüft. Beispiel:
Überwachung des Modellprozesses auf seinen "Bereit-Zustand":

```
DECLARE (BEREITBEGINN, SYSTEMZEIT) CLOCK;
...
SYSTEMZEIT = NOW; /* ABSPEICHERUNG DES AKTUELLEN ZEITPUNKTES */
IF SYSTEMZEIT <=BEREITBEGINN THEN
AT BEREITBEGINN ALL 2 SEC ACTIVATE ERFASSUNG;
                          ELSE GOTO ALARM;
FIN;
```

Zum durchgehenden Rechnerbetrieb sind folgende Anmerkungen zu machen: Erfolgt keine Überprüfung des Uhrzeitwertes durch den Benutzer, so hat dieser im Anschluß an die Einplanung keine Möglichkeit, zu erfahren, ob der Zyklus am selben oder erst am nächsten Tag gestartet wird.

An dieser Stelle ist zu erwähnen, daß in PEARL eine Einplanung mit einer Uhrzeit-Startbedingung einschließlich einer Datumsangabe nicht möglich ist, d.h. ein Auftrag kann frühestens einen Tag im voraus eingeplant werden, was i.a. aber wohl ausreicht und nicht als Einschränkung angesehen wird.

Beurteilung

PEARL enthält ein geeignetes Sprachelement zur Formulierung der Uhrzeit-Startbedingung bei konstanter und variabler Uhrzeit. Eine Überprüfung des Parameters dieser Bedingung ist unbedingt notwendig, um mit Sicherheit eine erfolgreiche Einplanung erzielen bzw. um unterscheiden zu können, für welchen Tag der Auftrag eingeplant wird. Die Programmierung der Startbedingung bereitet in diesem Fall keine Schwierigkeiten.

5.4.4 Zyklusstart nach Eintreffen eines Unterbrechungssignals

Realisierung

Diese Startbedingung wird mit 'WHEN Interruptausdruck' verwirklicht.

Implementierung

1. Beschränkung des Interruptausdrucks auf einfache Namen (APS1)
2. Zulassen von Interruptfeldern (Basis-PEARL).

Sprachuntersuchung

Die Beschränkung des Interruptausdrucks im APS1 auf einfache Namen hat zur Folge, daß eine Task mit nur einem Unterbrechungssignal verknüpft werden kann. Am Modellprozeß läßt sich dies an folgendem Beispiel verdeutlichen:

Zur Überwachung des Durchflusses bei Ansprechen sowohl von MAGNETVENTIL1 als auch von MAGNETVENTIL2 müssen für dieselbe Aufgabe zwei Tasks formuliert und zwei Operationen eingeplant werden:

```
WHEN MAGNETVENTIL1 ALL N ACTIVATE TA1;
WHEN MAGNETVENTIL2 ALL N ACTIVATE TA2;
```

Wesentlich gravierender wird dieses Problem bei einer sehr viel größeren Zahl von Unterbrechungssignalen, wie z.B. beim Stückprozeß-Modell "Hochregallager mit Materialtransportsystem".
Diese Anforderungen zeigen, daß es aus Gründen der Übersichtlichkeit und des Programmieraufwands erforderlich ist, Interruptfelder im obigen Ausdruck zuzulassen [47] , was in Basis-PEARL auch geschehen ist.

Das erwähnte Beispiel hat dann folgendes Aussehen:

```
WHEN MAGNETVENTIL(1:2) ALL N ACTIVATE TA;
```

Die damit entstandene Notwendigkeit für den Auftragnehmer, den Aktivierungsursprung, also den Auftraggeber festzustellen, wird ausführlich in [19] diskutiert.

Beurteilung

PEARL gestattet die Einplanung eines zyklischen Auftrags mit der Startbedingung "Eintreffen eines Interrupts" durch das Sprachelement 'WHEN Interruptausdruck'. Bei Zulassen von Interruptfeldern ist auch bei disjunktiver Verknüpfung+) mehrerer Interrupts die Einplanung gut realisierbar, nicht dagegen bei einer Beschränkung des Interruptausdrucks auf einfache Namen.

5.4.5 Zeitverzögerter Zyklusstart nach Eintreffen eines Unterbrechungssignals

Realisierung

Die Zusammensetzung der bekannten PEARL-Schlüsselworte WHEN und AFTER führt zu der Startbedingung 'WHEN Interruptausdruck AFTER Zeitdauerausdruck'.

Die damit verknüpfte Steueroperation wird erstmals nach dem sequentiellen Erfüllen der beiden Bedingungen ausgeführt:

1. Eintreffen eines Unterbrechungssignals
2. Ablauf der Verzögerungszeit nach Eintreffen des Signals.

Das Eintreten der WHEN-Bedingung ist Voraussetzung dafür, daß die AFTER-Bedingung erfüllt werden kann, ist also hier Vorbedingung.

+) auf die konjunktive Verknüpfung von Ereignissen wird ausführlich in [46] eingegangen.

Sie kann unabhängig von der nachfolgenden AFTER-Bedingung immer erfüllt werden und wird in [8] daher als "statische" Bedingung bezeichnet (statische (feste) Eintragung in der Unterbrechungsverwaltung). Im Gegensatz dazu wird die Bedingung 'AFTER Zeitdauerausdruck' in dieser zusammengesetzten Form dort als "dynamische" Bedingung bezeichnet (die Eintragung in die Zeitverwaltung erfolgt erst nach Eintreffen des Unterbrechungssignals).

Implementierung

Bei der Implementierung dieses Sprachelements treffen bezüglich des zulässigen Wertebereiches der Zeitdauer dieselben Aussagen wie in 5.4.2 zu, d.h. Zeitdauerwerte müssen entsprechend dem Rechnerbetrieb beschränkt werden, andernfalls wird eine Fehlerrückmeldung durch ein SIGNAL erzeugt.

Sprachuntersuchung

Ein typisches Beispiel für die Anwendung dieser Startbedingung am Modellprozeß ist die zyklische Erfassung der Drehzahl des Pumpenmotors nach dessen Hochlaufen:

```
DECLARE HOCHLAUFVERZOEGERUNG DURATION INITIAL (1 SEC);
SPECIFY EINTASTER INTERRUPT;

WHEN EINTASTER AFTER HOCHLAUFVERZOEGERUNG ALL 2 SEC ACTIVATE ERFASSUNG;
```

Beurteilung

PEARL läßt durch die Möglichkeit der Verknüpfung der beiden Schlüsselworte WHEN und AFTER eine direkte, einfach zu handhabende Einplanung mit dieser Startbedingung zu. Will man einen Einplanungs-Mißerfolg vermeiden, muß bei variabler Zeitdauer diese auf Plausibilität hin überprüft werden.
Bei nicht möglicher Einplanung wird dem Benutzer eine Fehlermeldung mitgeteilt.

5.5 Untersuchung und Beurteilung der Sprachelemente für Zyklus-Endebedingungen

5.5.1 Zyklusende durch explizites Löschen einer Einplanung

Für das Löschen aller Einplanungen eines Auftragnehmers - in diesem speziellen Fall das Löschen einer Einplanung der Operation ACTIVATE für eine Task ERFASSUNG - wurde in PEARL die Anweisung

[Schedule] PREVENT [taskname] ;

eingeführt.
Mit dieser Anweisung ist es möglich, das Zyklusende dadurch herbeizuführen, daß die Einplanung einschließlich evtl. gepufferter Aufträge vollständig gestrichen wird - optional unter Angabe von Steuerbedingungen: Zyklusende durch Löschen - also auch der Zyklusstart- und Wiederholungsbedingung.

Implementierung

1. Schedule bei PREVENT zulässig (bei Full-PEARL)
2. Schedule bei PREVENT nicht zulässig (beim APS1 und Basis-PEARL).

Sprachuntersuchung

Dieses Sprachmittel dient bei der Steuerung der Prozeßdatenerfassung für folgende Aufgaben:

1. Wurde bei der Einplanung einer Operation keine Endebedingung angegeben, kann eine zyklische Auftragserteilung durch das Löschen der Einplanung beendet werden.
   Es ist durch die Einführung dieser Anweisung erst möglich, die Zyklusstartbedingungen und das Zyklusende entkoppelt zu formulieren:

```
EINPLANUNG: TASK;
ALL 5 SEC ACTIVATE ERFASSUNG; /* EINPLANUNG OHNE ENDEBEDINGUNG */
END;
:
LOESCHEN: TASK;
PREVENT ERFASSUNG; /* ZYKLUSENDE DURCH LOESCHEN DER EINPLANUNG */
END;
```

2. Wurde bei der Operationseinplanung zwar eine Endebedingung angegeben, die aber aufgrund eines bestimmten Prozeß- oder Meßkanalzustandes nicht mehr einhaltbar ist, so kann das Zyklusende durch Löschen der Einplanung vorverlegt werden (s.a. 5.7 : Zyklusmodifikation).

Zwei denkbare Aufgabenstellungen können mit der PREVENT-Anweisung nicht unmittelbar gelöst werden:

- Einplanungen können nicht operationsspezifisch gelöscht werden (z.B. alle Einplanungen der Operation TERMINATE).
  Bei der Betrachtung der Anforderungen für Programme zur zyklischen Überwachung entfällt diese Aufgabenstellung durch die ausschließliche Verwendung der Operation ACTIVATE.
- Schedule-Elemente einer Einplanung können nicht gezielt gelöscht werden.
  Beispiel:
  WHEN VENTIL1 ALL 5 SEC, WHEN VENTIL2 ALL 2 SEC ACTIVATE ERFASSUNG;

  Soll beispielsweise nur das 1. Schedule-Element "unbekannt" werden, so kann mit der PREVENT-Anweisung das Löschen des 2. Elements sowie das Beenden eines evtl. "aktiven" Zyklus' dieses Elements nicht verhindert werden.
  Vor dem Löschen muß also festgestellt werden, welche der Schedule-Elemente, die erhalten bleiben sollen, "aktive" Zyklen besitzen.
  Nach dem Löschen (aller Schedule-Elemente) muß die Operation mit den noch benötigten Elementen neu eingeplant werden und zwar derart, daß die vorher "aktiven" Zyklen sofort wieder gestartet werden.

In diesem Abschnitt wurde für das erfolgreiche Löschen bisher vorausgesetzt, daß eine Operation für eine Task tatsächlich eingeplant war und das Zyklusende noch nicht erreicht worden ist.
Die Anwendungen haben bei der entkoppelten Formulierung von Start- und Endebedingung gezeigt, daß bezüglich der zeitlichen Reihenfolge des Löschens drei Fälle zu unterscheiden sind:

- 1 PREVENT wird ausgeführt, bevor eine Operation eingeplant wurde: PREVENT-Mißerfolg
- 2 PREVENT wird ausgeführt, nachdem eine Operation eingeplant und die Einplanung noch nicht gelöscht bzw. die Endebedingung noch nicht erreicht worden ist: PREVENT-Erfolg
- 3 PREVENT wird ausgeführt, nachdem die Einplanung bereits von anderer Seite vernichtet wurde: PREVENT-Mißerfolg.

In Fall 1 und 3 ist die PREVENT-Anweisung also überflüssig gewesen, was aber vor der Ausführung dieser Anweisung nicht bekannt war:

In PEARL gibt es bekanntermaßen keine Möglichkeit, auf Sprachebene festzustellen,

- ob eine Einplanung bereits erfolgt ist
- ob die Einplanung bereits gelöscht worden ist.

Während Fall 3 für den Anwender i.a. unkritisch ist (und dieser an manchen Programmstellen ein PREVENT "auf alle Fälle" vorsieht, eben weil er über den Schedule-Zustand nicht Bescheid weiß), führt Fall 1 zu nicht gewolltem Programmablauf. Dieser Fehler ist während der Modellprozeß-Programmierung bei wenig geübten Anwendern häufig aufgetreten. Die Fehlerlokalisierung war dabei bei mehrfach vorhandenen PREVENT-Anweisungen nicht immer einfach, eben weil keine Rückmeldung bei nicht erfolgreichem PREVENT erfolgt.

Daher läßt sich eine Forderung aufstellen, welche die Implementierung dieser Anweisung betrifft und in die PEARL-Sprachbeschreibung aufgenommen werden sollte (Liste der definierten SIGNALs in Anhang B [42]):

> Führt eine ohne Schedule formulierte PREVENT-Anweisung nicht zum Löschen zumindest einer Einplanung bzw. eines gepufferten Auftrags, so ist dem Benutzer eine Negativquittung in Form eines SIGNALs zu geben.

Für eine mit einem Schedule versehenen PREVENT-Anweisung trifft diese Forderung nicht zu, da dem Benutzer zur Zeit der Ausführung der Anweisung keine Negativquittung bei nicht erfolgreichem Löschen von Einplanungen mitgeteilt werden kann; über einen Erfolg kann erst bei Erfüllen der betreffenden Bedingung entschieden werden (!).

Im APS1[+)] war das Löschen unter Angabe von Bedingungen nicht zugelassen. Diese Einschränkung hat sich bei der Programmierung des Modellprozesses nicht ausgewirkt, da entweder das Zyklusende bei der Operationseinplanung angegeben werden oder die Einplanung ohne Steuerbedingungen, also unmittelbar mit PREVENT gelöscht werden konnte.

Beurteilung

Die Beurteilung des Sprachelements [Schedule] PREVENT [taskname] für das Löschen einer Einplanung zur zyklischen Auftragserteilung (Operation ACTIVATE) muß in vier Teile gegliedert werden:

+) in Basis-PEARL ebenfalls nicht

1. Der Schedule einer Einplanung besteht aus nur einem Element:
   Für das Löschen einer derartigen Einplanung ist das genannte Sprachmittel gut geeignet.
2. Der Schedule einer Einplanung besteht aus mehreren Elementen:
   - Löschen aller Elemente: Beurteilung s. 1.
   - gezieltes Löschen einzelner Elemente:
     Für diese Aufgabe ist das Sprachmittel aus den bereits genannten Gründen schlecht geeignet.
     Die Erweiterung müßte für diesen Fall dahin gehen, die Indices der zu löschenden Schedule-Elemente in der PREVENT-Anweisung mitangeben zu können.
3. Schedule bei PREVENT zulässig:
   Vorteilhaft ist, daß das Löschen (also auch das Zyklusende) eingeplant werden kann, über den Erfolg der Anweisung kann die auftraggebende Task aber keine Rückmeldung erfahren.
4. Schedule bei PREVENT nicht zulässig:
   Das Löschen erfolgt unmittelbar mit der Anweisung; vorteilhaft ist, daß dem Anwender der Erfolg oder Mißerfolg sofort mitgeteilt werden kann.

5.5.2 Zyklusende durch implizites Löschen einer Einplanung

In diesem Fall tritt das Zyklusende ein, ohne daß der Benutzer dabei die Absicht der Zyklus-Beendingung explizit ausdrückt. Dies trifft in PEARL dann zu, wenn eine Einplanung einer Operation für eine Task besteht, wenn dabei ein bzw. mehrere Zyklen "aktiv" sind und wenn dieselbe Operation an denselben Auftragnehmer - evtl. mit geändertem Schedule - erneut eingeplant wird. Ursache für das Ersetzen des bestehenden Schedules und das Beenden seiner "aktiven" Zyklen ist die task- (und operations)spezifische Verwaltung von Schedules.

Sprachuntersuchung

Eine Einplanung berührt den Taskzustand bekanntermaßen nicht, die Ausführung eines Auftrags bzw. noch vom ursprünglichen Schedule gepufferte Aufträge bleiben bestehen:
Das Zyklusende kann nicht definitiv bestimmt werden, ebensowenig der Zyklusbeginn des neuen Zyklus' bei der Startbedingung "Leerbedingung", da evtl. noch alte Aufträge vorher abgearbeitet werden .

Diese Art der Zyklusbeendigung wurde demgemäß immer nur dann angewandt, wenn der oben erwähnte Gesichtspunkt nicht berücksichtigt zu werden brauchte. Auf das Einfügen einer PREVENT-Anweisung - mit ihrem nicht unerheblichen Verwaltungsaufwand - konnte dann verzichtet werden.

Beurteilung

Die implizite Zyklusbeendigung ist dann geeignet, wenn die Zyklusbeendigung - und gegebenenfalls der erneute Zyklusstart - nicht zu einem genau definierten Zeitpunkt erfolgen muß[+)].

5.5.3 Zyklusende - "Leerbedingung"

Entsprechend der "Leerbedingung" für den sofortigen Zyklusstart ist es in PEARL möglich, auf die explizite Angabe einer Endebedingung bei der Einplanung zu verzichten. Das Einstellen der zyklischen Erfassung ist dann nur durch implizites oder explizites Löschen der Einplanung möglich.

Sprachuntersuchung

Diese Endebedingung, d.h. Weglassen einer explizit formulierten Zykusbeendigung, wurde dann eingesetzt, wenn die Zyklus-Zeitdauer bzw. der Zyklus-Endezeitpunkt bei der Einplanung nicht angegeben werden konnte.
Beispiel:
Überwachung des Modellprozesses auf seinen "Bereitzustand" bei unbekanntem Endezeitpunkt:

```
AT 8:00:00 ALL 2 SEC ACTIVATE ERFASSUNG;
```

Beurteilung

In PEARL hat der Anwender die Möglichkeit, bei der Einplanung einer Operation auf die Formulierung der Zyklus-Endebedingung zu verzichten: Einplanung und Zyklusendemitteilung (PREVENT bzw. implizites Löschen) können dadurch entkoppelt werden.

5.5.4 Zyklusende nach Ablauf einer Zeitdauer

In PEARL kann bei der Einplanung der zyklischen Auftragserteilung die maximale Zeitdauer eines Zyklus' mit der Endebedingung 'DURING Zeitdauerausdruck' programmiert werden.

+) Die Aussagen über das gezielte Löschen einzelner Schedule-Elemente eines Schedules entsprechen denen des letzten Abschnitts

Implementierung

Entsprechend dem verzögerten Zyklusstart müssen bei der Überprüfung des Ergebniswertes des Zeitdauerausdrucks die beiden Rechnerbetriebsarten unterschieden werden (s. 5.4.2). Die Reaktionen bei Überschreitung des Wertebereiches sind identisch.

Sprachuntersuchung

Bei der Anwendung dieser Endebedingung im Zusammenhang mit der Startbedingung 'AFTER Zeitdauerausdruck' traten dann Fehler auf, wenn nicht beachtet wurde, daß sich der Wert des Zeitdauerausdrucks nach dem Schlüsselwort DURING auf den Zyklusbeginn bezieht und nicht auf den Einplanungszeitpunkt.
Beispiel:
Zyklische Meßwerterfassung in den ersten zwei Minuten nach dem Einschalten der Kesselheizung

```
AFTER 5 SEC ALL 2 SEC DURING 2 MIN ACTIVATE  ERFASSUNG;
```

Das bedeutet, daß die Zyklusdauer 2 Minuten beträgt und nicht 1 Minute 55 Sekunden.

Neben diesem Gesichtspunkt ist auch hier der Ergebniswert eines Zeitdauerausdrucks vor der Einplanung auf Plausibilität hin zu überprüfen, will man immer eine erfolgreiche Einplanung erzielen.

Beurteilung

PEARL bietet ein gut geeignetes Sprachelement zur Realisierung einer Endebedingung, bei der angegeben wird, wie lange sich ein Zyklus im Zustand "aktiv" befinden darf. Diese Zeitdauer ist unabhängig vom Einplanungszeitpunkt, was insbesondere in Kombination dieser Endebedingung mit einem zeitverzögerten Zyklusbeginn zu beachten ist. Eine nicht realisierbare Einplanung infolge dieser Endebedingung wird dem Benutzer rückgemeldet (SIGNAL).

5.5.5 Zyklusende nach Erreichen einer Uhrzeit

In PEARL wurde hierfür ein spezielles Schlüsselwort eingeführt, die Bedingung hat im Zusammenhang mit ihrem Parameter folgendes Aussehen: UNTIL Uhrzeitausdruck.

Implementierung

Entsprechend dem Rechnerbetrieb sind hier ebenfalls zwei Fälle zu unterscheiden:
Die Prüfungen auf bestimmte Wertebereiche zur Feststellung, an welchem Tag der Zyklus beendet wird sowie die Fehlermeldungen entsprechen denen der Startbedingung "Uhrzeit" (s. 5.4.3).

Sprachuntersuchung

Die Kombination der Startbedingung WHEN mit der Endebedingung UNTIL hat sich als fehleranfällig erwiesen:
Aufgrund der Tatsache, daß die Startbedingung sowohl nach Eintreffen des Interrupts als auch nach Erfüllen der Endebedingung "Uhrzeit" "bekannt" bleibt, ist diese Kombination nur in Sonderfällen sinnvoll.
Beispiel:

```
WHEN START ALL 2 SEC UNTIL 18:00:00 ACTIVATE TA;
```

Trifft der Interrupt START vor 18 h ein, wird der Zyklus um 18 h dieses Tages beendet. Trifft er nach 18 h ein, so wird der Zyklus - abhängig vom Rechnerbetrieb - gar nicht mehr gestartet bzw. gestartet und erst um 18 h des nächsten Tages beendet. Der Endezeitpunkt ist also vom Zeitpunkt des Eintreffens des Unterbrechungssignals abhängig.

Wie bei der Startbedingung "Uhrzeit" ist auch bei der Endebedingung "Uhrzeit" die Angabe von Uhrzeit und Datum[+)] nicht möglich, beispielsweise für das Beenden einer Langzeitüberprüfung von Motorenprüfständen [51] .
Diese Aufgabe kann aber auf einfache Weise dadurch gelöst werden, indem die Uhrzeit in eine Zeitdauer-Endebedingung abgeändert wird.

Beurteilung

PEARL besitzt ein gutes Sprachelement zum Beenden eines Zyklus' zu einem absoluten Zeitpunkt am Tage bzw. am Tage nach der Einplanung. Für eine erfolgreiche Einplanung bzw. zur Unterscheidung des Tages des Endezeitpunkts ist eine Überprüfung des Parameters Uhrzeit erforderlich. Das Sprachelement ist nicht geeignet für eine Einplanung eines Endezeitpunkts, der mehr als einen Tag nach der Einplanung liegt. Für diesen speziellen Fall müßte die Erweiterung dahin gehen, als Parameter der Endebedingung neben der Uhrzeit das Datum mitangeben zu können - was aber eine Einführung eines Datentyps DATUM notwendig machen würde.

+) Für das Datum gibt es in PEARL keinen speziellen Datentyp. Die Angaben für Jahr, Monat und Tag werden in Daten des Typs FIXED abgespeichert und mit Standardprozeduraufrufen gesetzt und abgefragt.

5.5.6 Zyklusende nach Eintreffen eines Unterbrechungssignals

In PEARL kann ein Interruptbezeichner nur bei einer operationsauslösenden Steuerbedingung einer Einplanung stehen, d.h. ein Interrupt kann ausschließlich eine Startbedingung erfüllen. Es gibt kein entsprechendes Schlüsselwort analog zu DURING und UNTIL zur Angabe des Typs dieser Endebedingung.

Sprachuntersuchung

Die Formulierung der Endebedingung "Interrupt" wird durch implizites oder explizites Löschen der (vollständigen) Einplanung erreicht. Das bedeutet bei Vorhandensein mehrerer Elemente in einem Schedule, daß diejenigen Elemente, die man nicht löschen will, neu eingeplant werden müssen. Eine weitere Folge des Löschens ist natürlich außerdem die, daß eine zugehörige Startbedingung 'WHEN Interruptausdruck' "unbekannt" wird. Auch hier ist anschließend eine Neueinplanung der Operation notwendig und zwar immer dann, wenn die Zyklusendebedingung erfüllt wird (s. nachfolgende Beispiele).

Abhängig vom Typ der zugehörigen Startbedingung sind zwei Lösungen zu unterscheiden:

1. Startbedingung "Interrupt"

   Setzt man voraus, daß die Endebedingung des Typs "Interrupt" mit der Startbedingung des Typs "Interrupt" kombiniert wird - dies war bei den Anwendungen am Modellprozeß immer der Fall - ergibt sich folgendes Vorgehen:
   Es ist die Einführung einer zusätzlichen Task erforderlich, die nach Eintreffen des Ende-Interrupts beauftragt wird und in der dann die Operationseinplanung sofort gelöscht wird:

   Die Endebedingung wird in eine Startbedingung abgebildet.
   Beispiel: Überwachung des Wasserkessel-Heizungsstromes
   (Interrupt-Start- und Endebedingung)

```
SPECIFY (HEIZUNGEIN, HEIZUNGAUS) INTERRUPT;
...
WHEN HEIZUNGEIN ALL 5 SEC ACTIVATE UEBERWACHUNG;  /* EINPLANUNG */
WHEN HEIZUNGAUS ACTIVATE ENDETASK;  /* ZYKLUSENDE-EINPLANUNG */
...

UEBERWACHUNG: TASK;
...
/* ZYKLISCHE ERFASSUNG */
...
END; /* TASKENDE */

ENDETASK: TASK;
PREVENT UEBERWACHUNG; /* ENDE DER ERFASSUNG */
WHEN HEIZUNGEIN ALL 5 SEC ACTIVATE UEBERWACHUNG; /*NEUEINPLANUNG*/
END; /* TASKENDE */
```

2. Andere Startbedingungen

a) Lösung in Full-PEARL [42]:

Das Löschen der Einplanung ist mit folgender Anweisung möglich:

WHEN Interruptausdruck PREVENT [taskname] ;

Beispiel:
Beenden der Überwachung der Anlaufphase des Modellprozesses nach Abschalten der Heizung.

```
SPECIFY HEIZUNGAUS INTERRUPT;
...
EINPLANUNG: TASK;
ALL 5 SEC ACTIVATE ERFASSUNG;
/* EINPLANUNG DER OPERATION */

WHEN HEIZUNGAUS PREVENT ERFASSUNG; /* EINGEPLANTES LOESCHEN */
END;

ERFASSUNG: TASK;
...
/* ZYKLISCHE ERFASSUNG */
...
END; /* TASKENDE */
```

Vorteilhaft ist hier, daß auf eine zusätzliche Task verzichtet werden kann.

b) Lösung in Basis-PEARL:

siehe Startbedingung "Interrupt", Abschnitt 1.

Beurteilung

In PEARL kann eine Zyklus-Endebedingung des Typs "Eintreffen eines Interrupts" bei der Einplanung einer zyklischen Erfassung nicht direkt formuliert werden. Die Lösung der Aufgabe ist unter Erfüllung von Hilfsmaßnahmen möglich. Im Rahmen der Anwendung dieser Endebedingung erwiesen sich bei der Kombination mit der Startbedingung "Interrupt" - in APS1 und Basis-PEARL auch mit den übrigen Startbedingungen - folgende Punkte als nachteilig:

Nachteil 1: Schwere Programmlesbarkeit bzw. -verständlichkeit

Nachteil 2: Taskverschwendung (Verwaltungsaufwand - für zwei Anweisungen der Endetask)

Nachteil 3: Nach Eintreffen des Interrupts werden nicht sofort die anstehenden bzw. eingeplanten Aufträge gelöscht, sondern es wird erst die Endetask beauftragt - sie muß normalerweise vom Hintergrundspeicher transferiert werden - und erst dann wird das Löschen der Einplanung veranlaßt: Verzögerungszeit.

Die Realisierung mit den bisher vorhandenen Sprachmitteln ist nicht befriedigend.

Lösungsvorschlag zur Formulierung der "Zyklusbeendigung nach Eintreffen eines Interrupts"

Als Reaktion auf das Eintreffen von Unterbrechungssignalen sind in PEARL bisher ausschließlich Taskoperationen zugelassen:

- Erreichen von Taskzustandsänderungen
  (im Zusammenhang mit den Operationen ACTIVATE, TERMINATE, SUSPEND, CONTINUE und RESUME)
- Verhindern gepufferter bzw. zukünftiger Taskzustandsänderungen, resultierend aus Operationseinplanungen (im Zusammenhang mit PREVENT): Löschen von Einplanungen.

Will man das Ziel, also die Zyklusbeendigung nach Eintreffen eines Interrupts, auf Sprachebene formulieren, ohne über die beschriebenen Umwege zu gehen, muß das bisherige Prinzip aufgegeben werden.

Die Lösung der gestellten Aufgabe sieht dann entsprechend den Schlüsselworten DURING und UNTIL so aus, daß die Endebedingung durch ein Schlüsselwort und den Parameter 'Interruptausdruck' gebildet wird:

SCHLUESSELWORT Interruptausdruck

Hierbei sind zwei Alternativen möglich:

Alternative 1: Erweiterung der bisherigen Endebedingung UNTIL Uhrzeitausdruck zu

UNTIL { Uhrzeitausdruck / Interruptausdruck }

Vorteil:
Kein weiteres Schlüsselwort notwendig (mit einer Zunahme der Anzahl von Schlüsselworten steigt sowohl der Aufwand für das Übersetzen als auch für das Erlernen einer Sprache).

Nachteil:
Keine eindeutige Zuordnung von Schlüsselwort zum Typ der Endebedingung und dem Datentyp.
Beispiel:
Aus einer Anweisung wie
ALL 5 SEC UNTIL C ACTIVATE TA;
geht nicht mehr hervor, ob der Zyklus auf einen Interrupt mit dem Bezeichner C oder nach Erreichen einer Uhrzeit (Uhrzeitdate) beendet wird.

Dieser Nachteil kann durch geeignete Wahl von Bezeichnern für Interrupt- bzw. Uhrzeitdaten vermieden werden, wie beispielsweise INTERRUPTENDE oder UHRZEITENDE.

Alternative 2: Ergänzung der bisherigen Schlüsselworte UNTIL und DURING durch ein neues Schlüsselwort - z.B.

UNTILWHEN Interruptausdruck

Vorteil:
Eindeutige Zuordnung Schlüsselwort-Datentyp

Nachteil:
Einführung eines neuen Schlüsselwortes

Durch das Bestreben, die Anzahl der Schlüsselworte einer Sprache möglichst zu beschränken, wird hier die Alternative 1 der zweiten Variante vorgezogen. Das Schlüsselwort UNTIL steht also sowohl für Endebedingungen des Typs "Erreichen einer Uhrzeit" als auch für Endebedingungen des Typs "Eintreffen eines Interrupts".

Die Zyklus-Endebedingung "Eintreffen eines Interrupts" tritt im Zusammenhang mit den Zyklus-Startbedingungen in folgenden Kombinationen auf, wobei der Darstellung die gewählte Alternative zugrunde gelegt wird:

1. Sofortiger Zyklusbeginn nach der Einplanung ("Leerbedingung")

   ALL Zeitdauerausdruck UNTIL Interruptausdruck

   Nach Eintreffen des Interrupts erfährt dieses Schedule-Element eine Zustandsänderung von "bekannt" nach "unbekannt".

2. Startbedingung "Erreichen einer Uhrzeit"

   AT Uhrzeitausdruck $\left\{\begin{matrix}\text{ALL}\\\text{EVERY}\end{matrix}\right\}$ Zeitdauerausdruck UNTIL Interruptausdruck

   Das Eintreffen des Interrupts vor Erreichen des Startzeitpunktes hat keine Auswirkung (Zustandsänderung s. 1.).

3. Startbedingung "Ablauf einer Zeitdauer"

   AFTER Zeitdauerausdruck ALL Zeitdauerausdruck UNTIL Interruptausdruck

   Das Eintreffen des Interrupts vor Ablauf der AFTER-Zeitdauer hat keine Auswirkung (Zustandsänderung s. 1.).

4. Startbedingung "Eintreffen eines Interrupts"

   WHEN Interruptausdruck ALL Zeitausdruck UNTIL Interruptausdruck.

   Das Eintreffen des Startinterrupts vor Eintreffen des Endeinterrupts hat keine Auswirkung. Das Eintreffen des Endeinterrupts nach Eintreffen des Startinterrupts führt zur Einstellung des Zyklus' (Zykluszustand "passiv").

   Auch nach Beenden des Zyklus' bleiben die Startbedingung und das zugehörige Schedule-Element "bekannt", die Operation muß also nicht mehr neu eingeplant werden.

Das erwähnte Beispiel zur Überwachung des Wasserkessel-Heizungsstroms umfaßt hiermit folgende Vereinbarungen bzw. Anweisungen:

```
SPECIFY (HEIZUNGEIN, HEIZUNGAUS) INTERRUPT;
WHEN HEIZUNGEIN ALL 5 SEC UNTIL HEIZUNGAUS ACTIVATE UEBERWACHUNG;
...
UEBERWACHUNG: TASK;
...
END; /* TASKENDE */
```

5.6 Untersuchung und Beurteilung der Sprachelemente für die Zyklusunterbrechung bzw. -fortsetzung

Die Anforderung, einen Zyklus aufgrund eines bestimmten Prozeßzustandes befristet zu unterbrechen, kann in PEARL auf Sprachebene nicht gelöst werden. Der Grund ist, daß der Zykluszustand nach Erfüllen der Startbedingung bis zum Erfüllen der Endebedingung bzw. dem Löschen der Einplanung nicht beeinflußt werden kann.

Sprachuntersuchung

Die in PEARL vorhandenen Alternativen zur Realisierung dieser Anforderung sind folgende:

1. Die Zyklus-Unterbrechungsbedingungen werden durch das Löschen der Einplanung (mit Startbedingung gleich Fortsetzungsbedingung) realisiert.

   Dabei ist zu beachten, daß die Einplanung für die Fortsetzung des Zyklus' erst dann erfolgen darf, wenn die ursprüngliche Einplanung bereits gelöscht worden ist. Im anderen Fall wird infolge des Nichthäufens von Schedules die ursprüngliche Einplanung durch die "Fortsetzungs-Einplanung" sofort ersetzt, d.h. der Zyklus wird erstmals bei Erfüllen der Fortsetzungsbedingung "aktiv".

Beispiel:
Unterbrechung der Überwachung des Wasserkesseldrucks
während der Entnahmezeit

```
...
ALL 5 SEC ACTIVATE UEBERWACHUNG; /* SOFORTIGER ZYKLUSBEGINN */
WHEN TASTEEIN ACTIVATE HILFSTASK;   /* UNTERBRECHEN DER
                                       UEBERWACHUNG BEI TASTEEIN */
.
UEBERWACHUNG: TASK;
.
...
.
END;  /* TASKENDE */

.
.
HILFSTASK: TASK;
PREVENT UEBERWACHUNG;/* LOESCHEN DER EINPLANUNG: ZYKLUSENDE */
AFTER ENTNAHMEZEIT ALL 5 SEC ACTIVATE UEBERWACHUNG;
/* FORTSETZUNG DER UEBERWACHUNG NACH DER ENTNAHMEZEIT */
END; /* TASKENDE */
```

Die Nachteile dieser Lösung sind offensichtlich:
- Mangelnde Programm-Verständlichkeit
- Hoher Verwaltungsaufwand durch das Löschen und Neueinplanen; das Maskieren eines Zyklus' ist verwaltungsmäßig sehr viel einfacher zu realisieren, ähnlich dem Maskieren von Interrupts mit der PEARL-Anweisung 'DISABLE Interruptausdruck'.

2. Eine Abfrage erfolgt in den Tasks für die Aufgaben "Erfassung" und "Verarbeitung" anhand der Zulässigkeitstabelle, die angibt, ob bei einem betreffenden Prozeßzustand tatsächlich eine Erfassung bzw. Verarbeitung bestimmter Prozeßgrößen notwendig ist. Als nachteilig bei dieser Variante, die auch bei der Programmierung des Modellprozesses häufig verwendet wurde, hat sich erwiesen, daß das Einfügen zahlreicher Abfragen in den erwähnten Tasks sowie - bei erfolgter Zyklusunterbrechung - unnötige Taskaufrufe ("Taskleerlauf") erforderlich wurden.
Beispiel:
Erfassung mit Hilfe einer Zulässigkeitstabelle für 20 Analoggrößen des Modellprozesses in der Anlauf-, Arbeits- und Entnahmephase (s. Bild 3.9).

```
DECLARE (I,J) FIXED;
DECLARE ZULAESSIG(20,3) BIT(1); /* ZULAESSIGKEITSTABELLE FUER DIE
                                   DREI PHASEN */

/* '0'B1: KEINE ERFASSUNG, '1'B1: ERFASSUNG */
.
.
.
J = 2; /* ARBEITSPHASE */

FOR I FROM 1 TO 20 REPEAT
IF ZULAESSIG(I,J) = '1'B1 THEN CALL ERFASSUNG;
FIN;
                    END;
```

Vor jeder Erfassung einer Größe muß also die Notwendigkeit der Erfassung überprüft werden.

Beurteilung

PEARL besitzt kein spezielles Sprachmittel zum Unterbrechen bzw. Fortsetzen eines Zyklus'. Diese Aufgaben müssen über die beschriebenen Umwege erreicht werden.
Um die Nachteile beider Lösungen zu vermeiden, müßten in PEARL zwei neue Sprachelemente - ähnlich der PREVENT-Anweisung - eingeführt werden:

1. Ein Sprachelement zum Unterbrechen eines (mehrerer) "aktiven" Zyklus (Zyklen) einer Task TA und einer (mehrerer) Operation(en).
   Als Folge davon könnte ein neuer Zykluszustand "unterbrochen" eingeführt werden.
2. Ein Sprachelement zum Fortsetzen eines (mehrerer) "unterbrochenen" Zyklus (Zyklen) einer Task TA und einer (mehrerer) Operation(en).
   Zu entwickeln wäre also ein Sprachelement zur Realisierung des Zykluszustandswechsels "unterbrochen" nach "aktiv".

5.7 Untersuchung und Beurteilung der Sprachelemente für die Zyklus-Modifikation

Aus der Tatsache, daß mit einer Operationseinplanung alle Steuerbedingungen eines Schedules "bekannt" gemacht werden und mit dem Löschen der Einplanung diesselben "unbekannt" werden, geht hervor, daß der Typ einzelner Steuerbedingungen auf Sprachebene nicht änderbar ist. Eine Parameteränderung ist je nach Datentyp und Schedulezustand beschränkt möglich.

Sprachuntersuchung

Die Notwendigkeit einer Zyklus-Modifizierung ergab sich beim Modellprozeß insbesondere in Form von Zykluszeitänderungen sowie der Modifikation von Endebedingungen bei verändertem Prozeßablauf. Anfangs aufgetretene Fehler führten zu einer Einteilung der Modifikation entsprechend den Schedule- bzw. Zykluszuständen:

1. Die Operation ist noch nicht eingeplant

   Typänderung

   Die Modifikation des Typs einer bzw. mehrerer Steuerbedingungen einer noch nicht eingeplanten Operation ist nur dadurch möglich, daß man durch das Setzen vorgesehener Marken das Ausführen der ursprünglichen Einplanung verhindert und das Ausführen der Einplanung mit den geänderten Steuerbedingungen zuläßt.

   Parameteränderung

   - Interruptbezeichner: Keine Änderung möglich
   - Zeitdauervariable: Änderung durch eine Zuweisungs-Anweisung.

   Beispiel: Halbierung der Zykluszeit

```
DECLARE ZEIT DURATION;
...
ZEIT = 2 SEC;
...
ZEIT = ZEIT/2;
ALL ZEIT ACTIVATE UEBERWACHUNG;
```

2. Die Operation ist eingeplant, der zu ändernde Zyklus ist im Zustand "passiv"

   Es muß eine Neueinplanung der Operation mit den geänderten Steuerbedingungen bzw. geänderten Parametern erfolgen (die Neueinplanung beeinflußt aber bereits "aktive" Zyklen evtl. vorhandener weiterer Schedule-Elemente des Schedules).
   Beispiel:

```
DECLARE ZYKLUS DURATION;
ZYKLUS = 4 SEC;
...
AFTER 10 SEC ALL ZYKLUS ACTIVATE ERFASSUNG;
...
ZYKLUS = ZYKLUS/2;
AFTER 10 SEC ALL ZYKLUS ACTIVATE ERFASSUNG;
```

3. Die Operation ist eingeplant, der zu ändernde Zyklus ist im Zustand "aktiv"

Im Rahmen der Modellprozeßprogrammierung wurde eine Zyklusänderung bei diesem Zustand am häufigsten notwendig und zwar meist in Form einer Zykluszeitmodifikation, u.a. über die Bedienungsschreibmaschine (s.a. Datenmodell-Modifikation).
Je nach Art der Änderung sind folgende Realisierungen in PEARL möglich:

Typänderung

Im Zykluszustand "aktiv" kann eine noch "bekannte" Startbedingung des Typs "Interrupt" durch Löschen der Einplanung mit PREVENT und einer nachfolgenden Neueinplanung geändert werden, der "aktive" Zyklus wird aber dadurch "passiv". Er muß durch die Interrupt-simulations-Operation 'TRIGGER Interruptausdruck' wieder in den Zustand "aktiv" versetzt werden.
Hinsichtlich einer Typänderung der Endebedingung bei "aktivem" Zyklus sind zwei Fälle zu unterscheiden:

- Vorverlegung des Zyklusende-Zeitpunktes

Lösung in Full-PEARL:

In Full-PEARL ist eine Vorverlegung auf Sprachebene möglich:

Schedule PREVENT taskname ;

Beispiel:
Eine zyklische Überwachung, deren Ende für 17:00:00 eingeplant ist, soll beim nächsten Ansprechen des Grenzwertmelders eingestellt werden.

```
PROBLEM;
SPECIFY GRENZINTERRUPT INTERRUPT;

EINPLANUNG: TASK;
ALL 5 SEC UNTIL 17:00:00 ACTIVATE UEBERWACHUNG;
END; /* TASKENDE */

AENDERUNG: TASK;
WHEN GRENZINTERRUPT PREVENT UEBERWACHUNG;
END;
```

Der "aktive" Zyklus wird durch die PREVENT-Anweisung nur im Sinne der Vorverlegung des Zyklusendes beeinflußt.
Es erfolgt keine Verschiebung des Abtastzeitpunktes bei Ausführung der Anweisung.

Lösung in Basis-PEARL:

Eine Realisierung ist auf Sprachebene direkt nicht möglich. Sie muß durch Neueinplanen der Operation gelöst werden, entsprechend nachstehender Parameteränderung.

- Rückverlegung des Zyklusende-Zeitpunktes

  Sie erfolgt entsprechend nachstehender Parameteränderung.

Parameteränderung (Änderung der Zykluszeit bzw. des Parameters der Endebedingung):
Erforderlich für eine Modifikation ist eine Neueinplanung der Operation mit der geänderten Bedingung. Bei der Startbedingung des "aktiven" Zyklus' muß es sich dabei um eine "Leerbedingung" handeln, um eine Unterbrechung des Zyklus' zu vermeiden.
Durch die Neueinplanung ergibt sich eine Verschiebung des Abtastzeitpunktes zur Zeit T_1, die durch den Zeitpunkt des Ablaufs der Änderungstask bestimmt ist (s. Bild 5.5).

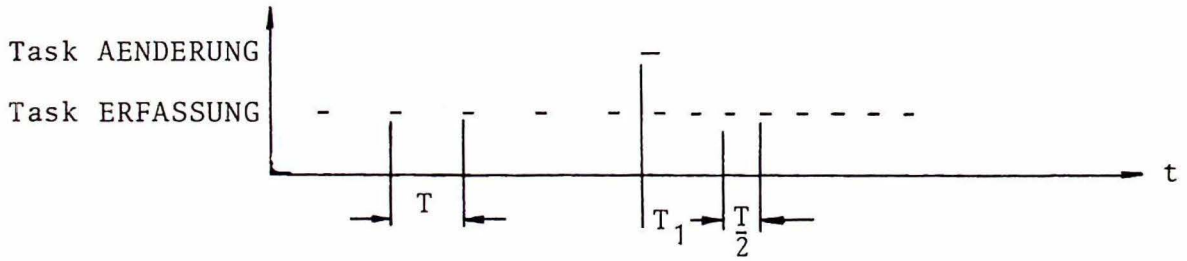


Bild 5.5: Änderung der Zykluszeit von T nach T/2

Beispiel: Halbierung der Zykluszeit

```
PROBLEM;
DECLARE ZEIT DURATION;
...
EINPLANUNG: TASK;
ZEIT = 2 SEC;
AFTER 10 MIN 30 SEC ALL ZEIT UNTIL 16:00:00 ACTIVATE ERFASSUNG ;
...
END; /* TASKENDE */

AENDERUNG: TASK;
ALL ZEIT/2  UNTIL 16:00:00 ACTIVATE ERFASSUNG ;
...
END; /* TASKENDE */
```

Schedules, die auch nach Erfüllen ihrer Endebedingung bzw. nach der Modifikation noch "bekannt" sein sollen - also ursprünglich eine Startbedingung "Eintreffen eines Interrupts" besaßen - müssen bei Erfüllen der Endebedingung wieder neu eingeplant werden.
Beispiel:
Während der dynamischen Grenzwertüberwachung der Wasserkesseltemperatur nach dem Einschalten der Heizung kann es u.U. notwendig werden, die Zykluszeit von einem Überwachungsprogramm aus zu ändern.

Beispiel:

```
DECLARE ZYKLUSZEIT DURATION;
SPECIFY (HEIZUNGEIN, HEIZUNGAUS) INTERRUPT;

EINPLANUNG: TASK GLOBAL;

ZYKLUSZEIT = 4 SEC;
WHEN HEIZUNGEIN ALL ZYKLUSZEIT ACTIVATE ERFASSUNG;
END; /* ENDE EINPLANUNG */

ERFASSUNG: TASK;
...
/* WASSERTEMPERATURERFASSUNG */
...
END; /* ENDE ERFASSUNG */

UEBERWACHUNG: TASK;

...
IF ZYKLUSZEIT < 2 SEC THEN GOTO ALARM;
ELSE BEGIN;
     ALL ZYKLUSZEIT/2 ACTIVATE ERFASSUNG; /* NEUEINPLANUNG */
     WHEN HEIZUNGAUS ACTIVATE EINPLANUNG; /* RUECKSETZEN DER
                                             ZYKLUSZEIT */
     END;
FIN;
...
END; /* ENDE UEBERWACHUNG */
```

Nach dem Start der Task EINPLANUNG und dem Eintreffen des Interrupts HEIZUNGEIN wird die Task ERFASSUNG zyklisch angestoßen. Während der Überwachung wird mit einem sofortigen Zyklusstart bei halber Zykluszeit die Operation neu eingeplant. Die ursprüngliche Startbedingung bzw. Zykluszeit wird durch das anschließende Neuaktivieren der Task EINPLANUNG erreicht.

Eine weitere Schwierigkeit ergibt sich, wenn eine Endebedingung vom Typ "Ablauf einer Zeitdauer" vorhanden ist:
Bei der Neueinplanung muß mittels der Standardprozedur NOW auf die aktuelle Rechneruhrzeit zugegriffen werden, um ein Ändern der Zyklusdauer zu vermeiden.

Beispiel:

Änderung der Zykluszeit während der Anlaufphase des Modellprozesses

```
PROBLEM;
DECLARE ZYKLUSZEIT DURATION, ANLAUFZEIT DURATION INITIAL (10 MIN),
        EINSCHALTZEIT CLOCK;
DECLARE NOW ENTRY RETURNS (CLOCK) GLOBAL; /* RECHNERUHRZEIT */

EINPLANUNG: TASK GLOBAL;

ZYKLUSZEIT = 10 SEC;
...
EINSCHALTZEIT = NOW;   /* ABSPEICHERUNG DER AKTUELLEN UHRZEIT */
ALL ZYKLUSZEIT DURING ANLAUFZEIT ACTIVATE ERFASSUNG;
...
END; /* ENDE EINPLANUNG */

ERFASSUNG: TASK;
...
/* PROZESSDATENERFASSUNG */
...
END;

MODIFIKATION: TASK GLOBAL;

DECLARE (AENDZEIT, RESTZEIT) CLOCK;
AENDZEIT = NOW;
IF ZYKLUSZEIT < 2 SEC THEN GOTO ALARM;
ELSE BEGIN;
     RESTZEIT = ANLAUFZEIT- (AENDZEIT-EINSCHALTZEIT);
     ALL ZYKLUSZEIT/2 DURING RESTZEIT ACTIVATE ERFASSUNG;
     END;
FIN;
END; /* ENDE MODIFIKATION */
```

Beurteilung

PEARL verfügt nicht über ein spezielles Sprachmittel zur Modifikation eines Zyklus'. Es hat sich herausgestellt, daß die Änderung entsprechend dem Schedule- bzw. Zykluszustand vorgenommen werden muß und sich je nach Zustand mehr oder weniger aufwendig gestaltet. Dies trifft insbesondere auf die wichtigste Anwendung - die Zykluszeitänderung im Zustand "aktiv" - zu:

Eine Änderung ist mit den vorhandenen Sprachmitteln (implizites bzw. explizites Löschen und einer Neueinplanung) dann sehr einfach realisierbar, solange die Wiederholungsbedingung nicht mit einer Startbedingung des Typs "Interrupt" oder einer Endebedingung des Typs "Zeitdauer" kombiniert ist. Für derartige Anwendungen sind die vorhandenen Sprachmittel schlecht geeignet. Für eine Zyklusänderung ist ein großer Aufwand notwendig, wie das letzte Beispiel zeigt. Der Grund ist der, daß bei der Einplanung einer Operation zumindest die Start- und Wiederholungsbedingung gemeinsam enthalten sind und nicht einzeln angegeben werden können - wie es bei dieser Aufgabenstellung erforderlich wäre.

Zusammenfassung

In der vorliegenden Arbeit wird im ersten Teil dargestellt, welche Voraussetzungen dem gewählten Weg der experimentellen Ermittlung von Sprachanforderungen und Spracheigenschaften zugrunde gelegt worden sind:

1. Es wird begründet, daß das bestehende Echtzeit-Betriebssystem gemäß den Anforderungen von PEARL anzupassen war. Dabei werden die Realisierungsmöglichkeiten, ihre Vor- und Nachteile sowie die gewählte Alternative vorgestellt und die Schwierigkeiten diskutiert. Es wird gezeigt, welche Betriebssystem-Module in einer Ablauforganisation vorhanden sein sollten, um den Adaptionsaufwand möglichst gering zu halten.
2. Im zweiten Abschnitt werden generelle Anforderungen an ein gegenständliches Modell zur Sprachuntersuchung zusammengefaßt. Es wird dargestellt, welchen Kriterien ein Modell des Grundtyps "Fließprozeß" und der Aufgabenstellung "Betriebsüberwachung" zu genügen hat. Die Beschreibung des Modells und der Prozeßvariablen zeigt, daß der Modellprozeß den Anforderungen entspricht und zur Sprachuntersuchung herangezogen werden kann.

Im zweiten Teil der Arbeit werden Sprachanforderungen erarbeitet und den vorhandenen Sprachelementen gegenübergestellt.

Zur Ermittlung der Sprachanforderungen werden am Modell typische Automatisierungsaufgaben abgeleitet. Für eine effektive Sprachuntersuchung werden sie zweckmäßigerweise in Teilaufgaben und untergeordnete Aufgabenelemente aufgeteilt. Als wichtige, charakteristische Teilaufgabe bei der Betriebsüberwachung von Fließprozessen wird insbesondere die Organisation der zyklischen Erfassung von Prozeßdaten betrachtet. Folgende Anforderungen ergeben sich hierbei:

- Einplanung der zyklischen Erfassung
- Zyklische Auftragserteilung
- Einstellung der Auftragserteilung
- Löschen von Einplanungen

Diese Funktionen werden anhand von Beispielen am Modellprozeß begründet und veranschaulicht.

Bei der Diskussion der Sprachelemente wird in einem ersten Schritt gezeigt, welche Sprachmittel in PEARL für den zyklischen Anstoß der Prozeßdatenerfassung zur Verfügung stehen. Daraus ergibt sich, daß die Sprachmittel für ein gezieltes Einplanen gut geeignet sind. Der Anwender ist von der Verwaltung der zyklischen Auftragserteilung befreit.

Bei der Untersuchung der Sprachelemente für die Steuerbedingungen erweisen sich diejenigen zur Programmierung der Start- und Wiederholungsbedingungen als gut, ebenso wie die Sprachelemente zur Formulierung der Zyklus-Endebedingungen - mit Ausnahme der "Zyklusbeendigung nach Eintreffen eines Interrupts". Um diese nur auf Umwegen lösbare Bedingung realisieren zu können, werden zwei Sprachvorschläge gemacht und ihre Vor- und Nachteile gegenübergestellt. Der gewählte Lösungsweg wird im Zusammenhang mit den verschiedenen Startbedingungen erörtert.

Anschließend wird gezeigt, daß in PEARL das Einstellen einer zyklischen Auftragserteilung bzw. Löschen einer Einplanung implizit und explizit ausgedrückt werden kann. Die Sprachmittel sind für diese Aufgabe dann gut geeignet, wenn ein Schedule nur aus einem Schedule-Element besteht bzw. wenn alle Elemente eines Schedules gelöscht werden sollen.
Ein gezieltes Löschen eines Schedule-Elementes ist auf Sprachebene nicht möglich.

Die Aufgabe, einen Zyklus befristet zu unterbrechen, kann in PEARL nicht direkt formuliert werden, sie kann aber auf das Löschen und erneutes Einplanen zurückgeführt oder durch Einführen einer Zulässigkeitstabelle erreicht werden. Um die Nachteile beider Lösungen zu vermeiden, müßten zwei neue Sprachelemente - ähnlich der PREVENT-Anweisung - eingeführt werden.

Die Zyklusmodifikation kann entsprechend dem Schedule- bzw. Zykluszustand durch eine Neueinplanung bzw. Zuweisung dann sehr einfach realisiert werden, solange die Wiederholungsbedingung nicht mit einer Startbedingung des Typs "Interrupt" oder einer Endebedingung des Typs "Zeitdauer" kombiniert ist. Es wird anhand von Beispielen verdeutlicht, daß sich die Zyklusänderung für diese speziellen Anwendungen mit den vorhandenen Sprachmitteln aufwendig gestaltet.

Literaturverzeichnis

[1] Lauber,R.: Prozeßautomatisierung I, Springer Berlin-Heidelberg-New York, 1976

[2] Kussl,V.: Prozeß-Automatisierungs-Sprache PAS1, Sprachbeschreibung, BBC System DP 1000, BBC Mannheim 1971

[3] RTL/2 language specification, ici central instrument research laboratory, England 1972

[4] SYSTEME PROCOL 2000, STERIA, Ref. 1 162 220/0039 00, Le Chesnay

[5] Timmesfeld, K.H. et. al.: PEARL-Vorschlag für eine Prozeß - und Experimentautomationssprache, Gesellschaft für Kernforschung mbH, PDV-Bericht KFK-PDV 1, Karlsruhe 1973

[6] Lauber,R.: Experimentelle Untersuchung von Spracheigenschaften der Prozeßrechnersprache PEARL anhand von Modellprozessen, Applied Computer Science 3, Hanser Verlag (1976), 13-30

[7] Wiedenmann,R.: Anforderungen des ASME-PEARL-Subsets an eine Ablauforganisation sowie ein Entwurf der Anpassungsrealisierung, ASME-Interne Notiz S-015, Stuttgart 1974

[8] Wiedenmann,R.; Zeh,A.: Interrupt-Behandlung und Zeitbeauf - tragung von Tasks, ASME-Interne Notiz S-018, Stuttgart 1974

[9] Helfert,M.; Wiedenmann,R.: Realisierbarkeit des PEARL-Tasking an der AEG 60-50, ASME-Interne Notiz S-004, Stuttgart 1973

[10] Wiedenmann,R.: Beschreibung der Schnittstelle PEARL-Compiler/ Assembler(AEG 60-50), Gesellschaft für Kernforschung mbH, Karlsruhe, Nr. E 65

[11] Richtlinien-Entwurf VDI/VDE 3552 "Leistungskriterien von Prozeßrechensystemen", Berlin, Beuth-Verlag, 1975

[12] Brunner,P et. al.: Universelles PEARL-Betriebssystem, Gesellschaft für Kernforschung mbH, Karlsruhe KFK-PDV 55, 1976

[13] Handbuch Betriebssystem MARVIS, AEG-Telefunken Berlin, E 524/V3 13.9 1270, Berlin 1971

[14] Bautz,J.: Auftragsverkehr über einen generellen Auftragspuffer bei Prozeßrechensystemen, GI-Tagung Karlsruhe (1972), Lecture Notes in Computer Science, Springer Verlag, Vol 1, S. 128

[15] Eichenauer,B.: Dynamische Prioritätsvergabe an Tasks in Prozeßrechensystemen, Diss. Universität Stuttgart 1975

[16] Dijkstra, E.W.: Cooperating Sequential Processes, In Programming Languages (Editor F. Genuys) London, Academic Press 1968

[17] Rieder,P.: Prozeßzustände bei Echtzeit-Programmiersprachen, Tagung Prozeßrechner 1974, Karlsruhe, Lecture Notes in Computer Science 12 (1974) 413-424, Springer 1974

[18] Funktionelle Beschreibung von Prozeßrechner-Betriebssystemen, VDI/VDE-Entwurf, DK 681.32.o6(o83.132), März 1976

[19] Helfert,M.: Eine Untersuchung der Anforderungen an eine Prozeßprogrammiersprache bei der Automatisierung von Stückprozessen unter besonderer Berücksichtigung von PEARL,Dissertation Universität Stuttgart 1978

[20] Püntzner,W.: Untersuchung von Meßwertgebern und Meßverfahren für den Modellprozeß "Betriebsüberwachung eines Fließprozesses", 1. Semesterarbeit am Institut für Regelungstechnik und Prozeßautomatisierung, Universität Stuttgart, 1974

[21] Lauber,R.; Zeh,A.: Experimental Investigation of Process Control Language Features by Process Models, Third Europ. Seminar on Realtime Programming, Ispra, Italy 1973

[22] Lauber,R.; Neff,O.: Vorlesungen bzw. Seminare über Prozeß - automatisierung II, Universität Stuttgart, WS 1976/77

[23] Schoeffler, J.D.: The development of process control software, Spring Joint Computer Conference(1972), 9o7-914

[24] Görlich, .: Zu Problemen der Meßtechnik und Meßdatenverarbeitung für die Automatisierung, msr 16(1973), Heft 6

[25] Steusloff,H.: Automatische Systemsicherung in prozeßrechnergeführten Systemen, DFG Bericht Sy 6/3, Karlsruhe (1973)

[26] Hotes,H.: Digitalrechner in technischen Prozessen, Walter de Gruyter, Berlin 1967

[27] Müller,C.et. al.: Prozeßüberwachung mit dem Programmsystem MADAM, Siemens-Zeitschrift 45(1971), Heft 1o

[28] Prozeßautomatisierung mit dem Programmpaket ARSI, Druckschrift der Fa. AEG-Telefunken

[29] BICEPS Supervisory Control, Summary Manual, GE Co., Phoenix, Arizona, 1969, GET 3559A, S. 129

[30] Winkler, J.F.H.: Kostenbetrachtungen bei der Realisierung von Datenstrukturen, Gesellschaft für Kernforschung mbH, Karlsruhe KFK-PDV 25, 1974

[31] Emge,R.: Maschinennahe Struktur von Standard-Software und deren Behandlung mittels der höheren Sprache SL3, Applied Computer Science 3, Hanser Verlag, 1976

[32] Schmider,K.: Programmierung von Automatisierungsaufgaben der Prozeßüberwachung in PEARL, Diplomarbeit am Institut für Regelungstechnik und Prozeßautomatisierung, Universität Stuttgart, 1977

[33] Späth,H.: Algorithmen für elementare Ausgleichsmodelle, Oldenburg Verlag, 1973

[34] Kendziorra,J.: Analytische Approximation nichtlinearer Wandlerkennlinien für die elektronische Linearisierung, msr 17(1974) 1

[35] Spal,J.: Glättungsalgorithmen für die digitale Meßwerterfassung, Regelungstechnik und Prozeßdatenverarbeitung (197o), Heft 9, 39o-395

[36] Engels,H.: Iterative Glättung von Meßdaten, Angewandte Informatik Nr. 6, 1972, S. 257-262

[37] Glättungsalgorithmus GLT, Siemens Programmsystem MADAM, wissenschaftlicher Bericht 25o ooo 4.7o 1559 (1971)

[38] Wendelin,R. et.al.: Datenverknüpfung im Prozeßrechner mit Hilfe von Entscheidungstafeln, Angewandte Informatik Heft 11 (1971), 493-499

[39] Lauber,R.: Experimentelle Ermittlung der Leistungsfähigkeit von Prozeßrechnern bei ereignisgesteuerter Programmverarbeitung Fachtagung Prozeßrechner 1977, Informatik-Fachbericht 7, Springer Verlag (1977)

[40] PEARL Subset for Avionic Applications, Language Description, June 1976, ESG mbH, München

[41] Entwurf DIN 66 200: Betrieb von Rechensystemen, DK 681.31/.34:001.4; Begriffe Auftragsabwicklung

[42] FULL-PEARL Language Description, Gesellschaft für Kernforschung mbH, Karlsruhe, KFK-PDV 130, 1977

[43] Basic PEARL Language Description, Gesellschaft für Kernforschung mbH, Karlsruhe, KFK-PDV 120, 1977

[44] Heller,G. et. al.: Prozeß-FORTRAN 75, eine Erweiterung von FORTRAN für Prozeßrechner-Anwendungen; Vorschlag des Arbeitskreises "Prozeß-FORTRAN" der VDI/VDE Gesellschaft für Meß- und Regelungstechnik (1975)

[45] Koch,G. et. al.: Bewertung der Beschlüsse des PEARL-Subset-Arbeitskreises bzgl. des Scheduling und Bereinigungsvorschläge, Studie Nr. 41/76, BBC Mannheim

[46] Ghassemi,A.: Untersuchung der Eignung der Prozeßprogrammiersprache PEARL zur Automatisierung von Folgeprozessen, Dissertation Universität Stuttgart 1978

[47] Helfert,M.: Änderungsvorschlag zu PEARL, SAK Ä-Nr. 21/76, Gesellschaft für Kernforschung mbH, Karlsruhe (1976)

[48] Rössler,R.: Betriebssystem-Strategien zur Bewältigung von Zeitproblemen in der Prozeßautomatisierung, Dissertation Univers. Stuttgart 1978

[49] Püntzner,W.: Programmierung der Überwachung von Analog - und Binärgrößen des Modellprozesses "Betriebsüberwachung" in PEARL, Diplomarbeit am Institut für Regelungstechnik und Prozeßautomatisierung, Stuttgart 1976

[50] Khuong Long,T.: Dynamische Grenzwertkontrolle des Modellprozesses "Betriebsüberwachung", Studienarbeit am Institut für Regelungstechnik und Prozeßautomatisierung, Stuttgart 1977

[51] Bühler,F.: Einsatz eines Prozeßrechners für Langzeit - prüfungen bei Zuverlässigkeitsuntersuchungen, BOSCH Techn. Berichte 3-Heft 7 - April 1972

Wiedenmann, R.
Untersuchung der Eignung der Prozeßrechnersprache PEARL zur Programmierung von Automatisierungsverfahren der zyklischen Prozeßdatenerfassung

Kernforschungszentrum Karlsruhe GmbH
PDV-Bericht, KfK-PDV 164, November 1978
131 Seiten, 31 Abbildungen, 1 Tabelle, 51 Literaturstellen

Die Voraussetzungen der Sprachuntersuchung werden zusammengestellt und diskutiert. Es wird gezeigt, wie die Anforderungen von PEARL an ein Echtzeit-Betriebssystem durch die Anpassung des Betriebssystems des Prozeßrechners AEG 60-50 erfüllt werden.
Das zur Durchführung der Sprachuntersuchung gewählte gegenständliche Modell eines Fließprozesses mit der Aufgabenstellung Betriebsüberwachung wird dargestellt. Die charakteristischen Automatisierungsaufgaben zur Ermittlung der Sprachanforderungen werden an diesem Modell abgeleitet. Es wird diskutiert, wie die Anforderungen der zyklischen Prozeßdatenerfassung mit Sprachmitteln von PEARL gelöst werden können.

Wiedenmann, R.
Untersuchung der Eignung der Prozeßrechnersprache PEARL zur Programmierung von Automatisierungsverfahren der zyklischen Prozeßdatenerfassung

Kernforschungszentrum Karlsruhe GmbH
PDV-Bericht, KfK-PDV 164, November 1978
131 Seiten, 31 Abbildungen, 1 Tabelle, 51 Literaturstellen

Die Voraussetzungen der Sprachuntersuchung werden zusammengestellt und diskutiert. Es wird gezeigt, wie die Anforderungen von PEARL an ein Echtzeit-Betriebssystem durch die Anpassung des Betriebssystems des Prozeßrechners AEG 60-50 erfüllt werden.
Das zur Durchführung der Sprachuntersuchung gewählte gegenständliche Modell eines Fließprozesses mit der Aufgabenstellung Betriebsüberwachung wird dargestellt. Die charakteristischen Automatisierungsaufgaben zur Ermittlung der Sprachanforderungen werden an diesem Modell abgeleitet. Es wird diskutiert, wie die Anforderungen der zyklischen Prozeßdatenerfassung mit Sprachmitteln von PEARL gelöst werden können.

Wiedenmann, R.
Analysis of the Capability of the Realtime Programming Language PEARL to Perform Cyclic Process Data Acquisition.

Kernforschungszentrum Karlsruhe GmbH
PDV-Report, KfK-PDV 164, November 1978
131 pages, 31 figures, 1 table, 51 references

The prerequisites of the language analysis are compiled and discussed. It shows, how the realtime operating system of the process computer AEG 60-50 is adapted to meet the requirements of PEARL.
The physical model of a continuous process with the task of process control selected for execution of the language analysis is described. The characteristic automation functions for investigation of the language requirements are deduced from this model.
The features of the language PEARL, fulfilling most of the requirements of cyclic process data acquisition, are discussed.

Wiedenmann, R.
Analysis of the Capability of the Realtime Programming Language PEARL to Perform Cyclic Process Data Acquisition.

Kernforschungszentrum Karlsruhe GmbH
PDV-Report, KfK-PDV 164, November 1978
131 pages, 31 figures, 1 table, 51 references

The prerequisites of the language analysis are compiled and discussed. It shows, how the realtime operating system of the process computer AEG 60-50 is adapted to meet the requirements of PEARL.
The physical model of a continuous process with the task of process control selected for execution of the language analysis is described. The characteristic automation functions for investigation of the language requirements are deduced from this model.
The features of the language PEARL, fulfilling most of the requirements of cyclic process data acquisition, are discussed.