Der Einsatz von PEARL für eine Echtzeitsimulation auf einem Prozessrechner in Verbindung mit einem Hybridrechner

Dipl.Ing. Claudia Schmidt

1 Aufgabenstellung

1.1 Allgemeine Beschreibung des Problems

Das Projekt beschäftigt sich mit der Entwicklung neuer Antriebsregelkonzepte für Personenaufzüge. Dabei steht neben der Verbesserung von Komfort und Einfahrgenauigkeit die Realisierung einer digitalen Antriebsregelung zur Aufgabe. Der Vorteil einer digitalen Lösung gegenüber einer herkömmlichen liegt in der Zusammenfassung von Antriebsregelung und Ablaufsteuerung, die heute schon weit verbreitet mit Hilfe der Mikroprozessortechnik erfolgt.

Im Zuge einer digitalen Reglerimplementierung sollen gleichzeitig Parameterschwankungen der Aufzugsstrecke (Eigenfrequenzen,Dämpfung) und andere Einflüsse berücksichtigt werden, was bei einer analogen Ausführung nur schwer möglich ist.

Einen Teil des Projektes bildet die Simulation unterschiedlicher Antriebsregelungen. Dabei sollen im Wesentlichen folgende Punkte untersucht werden:

- Einflüsse der in der Regelung nicht berücksichtigten Modellabschnitte,
- Einflüsse der sich während des Betriebes verändernden Parameter (Last,Wegänderung),
- Störungen (Bremse,Schienenstöße),
- Anforderungen an den elektrischen Antriebskreis,
- Anforderungen an den Prozessor,die Schnittstelle und die Messerfassungssysteme.

Bild 1 zeigt eine Übersicht über das physikalische Modell einer geregelten Aufzugsanlage. Hierbei ist die Sicherheitskontrolle nicht enthalten.

1.2 Simulationsmethoden

Bei der Wahl des Simulationskonzeptes wurde darauf geachtet, nahe der Realisierung zu liegen, um eine hohe Sicherheit in der Voraussage des Verhaltens der Anlage zu erzielen. Es wurde eine Echtzeitsimulation gewählt, die den Vorteil

- der Eingriffsmöglichkeit während des Prozesses,
- der Aussage über Laufzeiten der Prozeßglieder,
- der Nachbildung paralleler Prozesse bei der Aufzugsstrecke,
- der Einbeziehung der Schnittstellen, die bei der Realisierung erforderlich sind (A/D - D/A - Wandlung)

beinhaltet.

Die Nachbildung des Streckenmodells wurde als kontinuierliches, dynamisches System auf einem Hybridrechner ausgeführt. Dieser Rechner vereinigt analoge und digitale Rechenelemente, so daß eine Berücksichtigung nicht linearer Elemente (Reibung,Bremsverhalten) möglich ist. Der Motion Control - Teil (Sollwertbildung,Regelung) wurde in einen Prozessrechner verlegt. Zum Test der Abtastzeiten hilft die Verwendung der Zeitraffung und -dehnung, die beim Hybridrechner möglich ist. Die Grenze dieser Zeitmanipulation wird durch die Offsetdrift der Integrierer bei langen Prozeßlaufzeiten gesetzt.

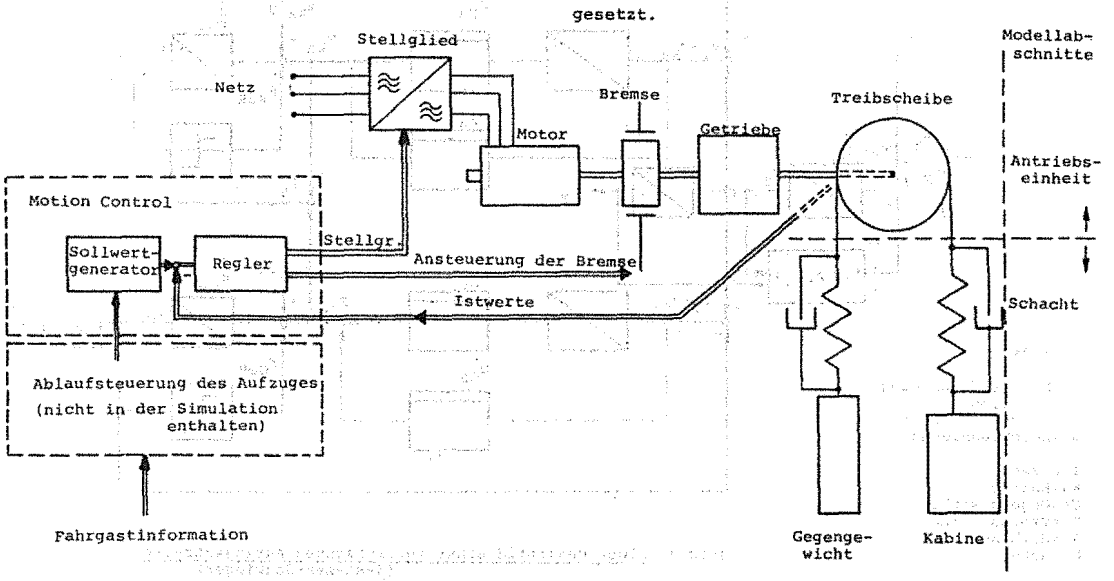

Bild 1 Übersicht des physikalischen Modells

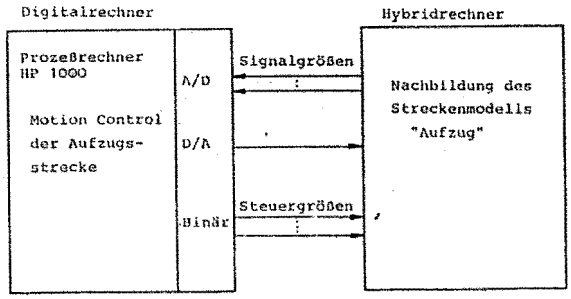

Bild 2 Simulationsaufbau

2 Realisierung der Simulation

2.1 Das Streckenmodell auf dem Hybridrechner

Das Modell des Aufzuges wird unterteilt in einen elektrischen und einen mechanischen Teil. Der mechanische Teil ist ein gekoppeltes Mehrmassensystem,der je nach Ausführung und Vereinfachung von einem Differentialgleichungssystem 6.Ordnung beschrieben werden kann (Antriebseinheit - Kabine - Gegengewicht). Der elektrische Teil wird als drehmomentgeregelter "schwarzer Kasten" angesehen und als VZ1-Glied 1. Ordnung (plus Totzeit) realisiert.
Nichtlinearitäten und Störgrößen (Reibungs- und Bremsverhalten) können mit Hilfe von Funktionsgebern nachgebildet werden. Es besteht auch die Möglichkeit über den Prozeßrechner Kennlinien von Störgrößen zu realisieren. Bei dem jetzigen Stand der Untersuchung wurde jedoch nur von der Sign-Funktion (Haftreibung) Gebrauch gemacht. Bild 3 zeigt das auf dem Hybridrechner realisierte Blockschaltbild wie es aus dem Bereich der Regelungstechnik üblich ist.

2.2 Reglerimplementierung auf dem Prozeßrechner

2.2.1 Reglerrealisierung

Auf dem Prozeßrechner erfolgt die Sollwertberechnung, Regelung und Protokollierung (mit anschließender Auswertung). Dabei wurden alle 3 Funktionen als eigenständige Programmteile gesehen, die miteinander kommunizieren. Das hat den Vorteil, daß bei Änderung des Regelkonzeptes nur der Reglerteil berührt ist, Sollwertberechnung und Messauswertung aber erhalten bleiben.

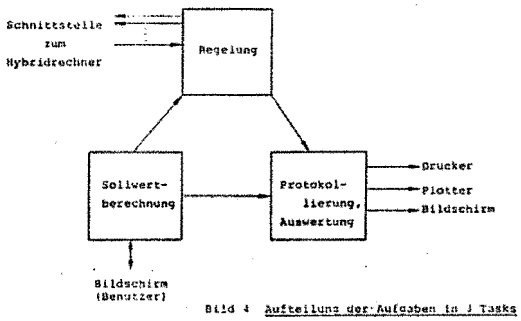

Bild 4 Aufteilung der Aufgaben in 3 Tasks

Bei der Auslegung des Reglers geht man von einer diskreten Strecke aus, die in Abhängigkeit der Abtastzeit und physikalischen Parameter ihre Pole ändert.
Folgende Anforderungen an die Regelung sind zu berücksichtigen:

- ein zeitoptimales Führungsverhalten,
- daß die für den menschlichen Körper spürbaren Grenzwerte der Beschleunigung und der Beschleunigungsänderung auch bei betriebsnormalen Störungen (Bremse,Haftreibung) nicht überschritten werden,
- Vermeidung eines Überschwingen des Wegistwertes,
- daß die Stellgröße beschränkt ist,
- eine hohe Positionsgenauigkeit,
- geringer Aufwand bei der Messerfassung.

Der Einfachheit halber wurde bei der Auslegung von einer Strecke 3. Ordnung ausgegangen, d.h. die Treibscheibe ist mit dem Gegengewicht und der Kabine starr gekoppelt. Das hat nur dann seine Gültigkeit, wenn die Anregung des

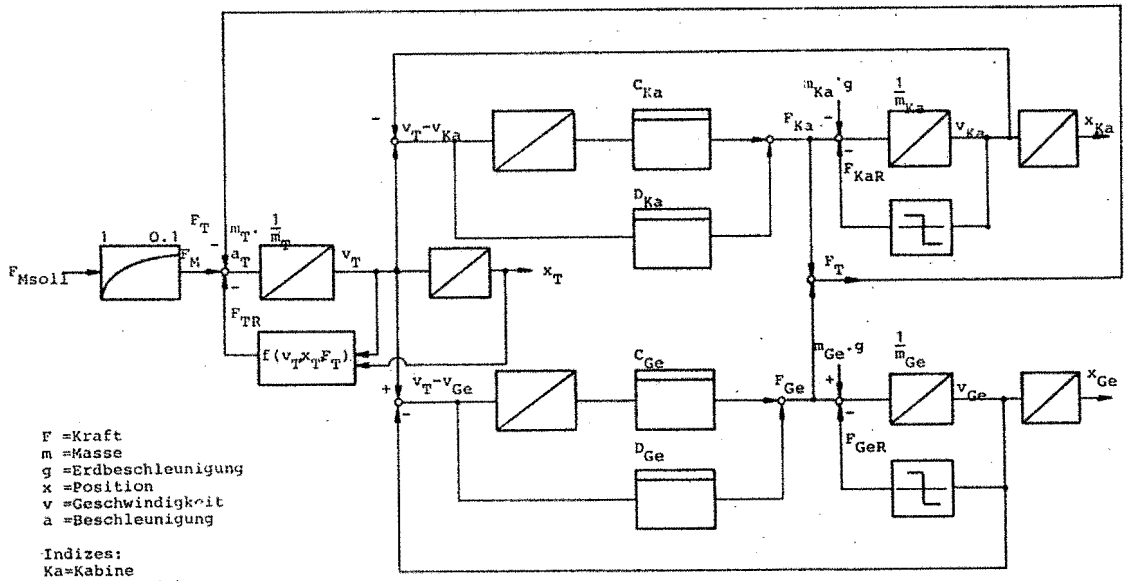

Bild 3 Blockschaltbild einer vereinfachten Aufzugsstrecke (3-Massen-Schwinger)

Systems unter der Eigenfrequenz des Schachtes bleibt. Hierbei wird gerade der letzte Punkt der Anforderungen an die Regelung erfüllt.

Die Auslegung der Regelung geschieht auf Grundlage verschiedener Regelverfahren:

Kaskadenregelung (Nachteil: alle 3 Zustände - x,v,a - müssen erfaßt werden)

Kompensationsregler für den Wegregelkreis (Nachteil: kein Einfluß auf andere Zustände,z.B. v,a)

Zustandsregler (ausbaufähig, jedoch großer Rechenaufwand)

Auf die Regelverfahren wird hier nicht weiter eingegangen. Es wird nur ein Beispiel genannt, um die anschließende Problematik mit der Programmiersprache PEARL besser zu verdeutlichen.

Zu Testzwecken wurde zunächst ein Kaskadenregler aufgebaut, wie er auch aus der kontinuierlichen Regelungstechnik bekannt ist.

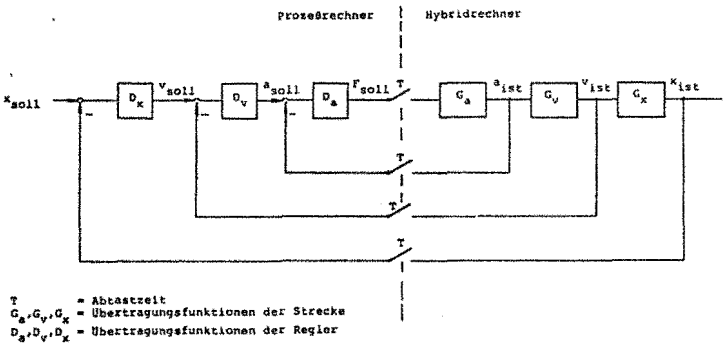

Bild 5 Vereinfachtes Blockschaltbild der Kaskadenregelung

Der gesamte Regelalgorithmus für die einzelnen Regelkreise wurde sequentiell im Rhythmus der Taktzeit T abgearbeitet (Bild 6).

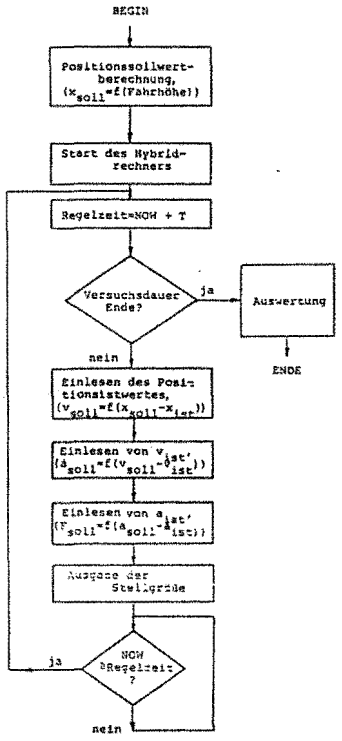

Bild 6 Übersicht der Datenfluß-Software

Alternativen zur Realisierung des Kaskadenreglers

Gedacht war auch an eine Kaskadenregelung mit unterschiedlichen Abtastzeiten für die einzelnen Regelkreise. Da der Beschleunigungsregelkreis sehr schnell auf Laständerungen an der Treibscheibe reagieren muß,ist es sinnvoll diesen Regelkreis erheblich schneller zu gestalten als den Drehzahlregelkreis.
Wählt man die Abtastzeit des äußeren Regelkreises um ein Vielfaches des inneren, so darf die Rechenzeit des gesamten Reglers die Abtastzeit des inneren Kreises nicht überschreiten. Das Problem ist einfach mit einer sequentiellen Programmgestaltung zu lösen. Soll der Beschleunigungsregler immer im Eingriff sein (Rechenzeit des Beschleunigungsregelkreises= Taktzeit des inneren Reglers) und nur von der überlagerten Regelung unterbrochen werden, so könnte theoretisch das Reglerprogramm mit Hilfe zweier Tasks und Semaphoren gestaltet werden.

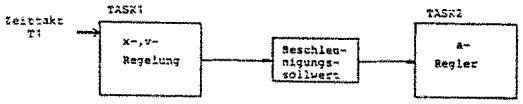

Bild 7 Aufteilung der Regelung in 2 Tasks

Die Starttask(Task 1 = Geschwindigkeits- und Positionsregler) aktiviert Task2 (Beschleunigungsregler). Beide Tasks greifen auf den Beschleunigungssollwert zurück. Die Task2 wartet bis die Semaphore "Sollwert_a" frei ist. In der Starttask wird der Sollwert der Beschleunigung berechnet und die Semaphore frei gegeben. Task2 kann nun die Stellgröße berechnen.
Da es sich bei der Task2 um eine Endlosschleife handelt, in der keine Wartezustände auftreten (bei analoger Ein-/Ausgabe wird direkt das entsprechende Interface angesprochen), besteht (bei Tasks gleicher Priorität) für das PEARL-System keinen Anlaß zum Taskwechsel. Somit wird die Starttask nie wieder aktiviert.

Programmbeispiel:

```
        :
        :

DECLARE
     :
     :
   Sollwert_a    SEMA PRESET(1)
     :

START_Regelung : TASK;
     :
  ACTIVATE TASK2;
  WHILE Versuchsdauer GT Dauer
  REPEAT
    :
    REQUEST Sollwert_a;
    Beschleunigungssollwert:= .....
    RELEASE Sollwert_a;
    :
    AFTER T1 RESUME;
    Dauer := NOW-Versuchsbeginn;
  END;
  TERMINATE TASK2;
```

```
TASK2 : TASK;
DECLARE
     :
     :
   laufende Berechnung BIT(1)
                INITIAL('1'B);
WHILE laufende_Berechnung
REPEAT
  :
  REQUEST Sollwert_a;
  Stellgröße := ........
  RELEASE Sollwert_a;
  :
  SEND Roh_Stellgröße TO
          Stellgrößenausgang;
END;
```

Das Problem wird gelöst, indem man die Tasks unterschiedlich priorisiert. So erhält die Starttask eine höhere Priorität als Task2. Sobald die Wartezeit in der Starttask abgelaufen ist, wird der Prozeß in der Task2 unterbrochen. Der Positions- und Geschwindig-

keitsregelalgorithmus berechnet den neuen Beschleunigungssollwert. In der verbleibenden Zeit (T1 - Rechenzeit des überlagerten Reglers) arbeitet die Task2.

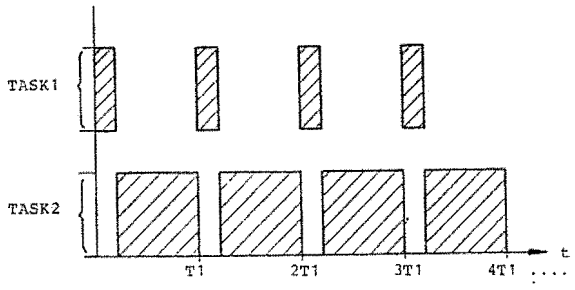

Bild 8 Zeitschema der Taskbearbeitung

Infolge des hohen Verwaltungsaufwandes kann bei diesem Betriebssystem ein Taskwechsel nur alle 2o ms stattfinden. Damit die reale Abtastzeit nicht variiert, wird T1 als ein Vielfaches von 2o ms gewählt. Nachteilig ist ebenfalls die mangelnde Vorausberechenbarkeit der Abtastzeit des inneren Regelkreises (= Rechenzeit der Task2).

2.2.2 Protokollierung und Auswertung

Sinnvoll ist es neben dem Programm die Protokollierung und Auswertung vorzunehmen. Dabei wird die Auswertung mit niedriger Priorität während der Wartezeit der Reglertask durchgeführt.
Zur Zeit erfolgt die Auswertung nach der simulierten Fahrt. Im Anschluß einer Fahrt werden die Messwerte geplottet. Dazu benötigt man Standard-Grafiksoftware der Fa. HP. Hierbei wird von der Möglichkeit der Einbettung externer Routinen (mit Schnittstellenprüfung) in das vorhandene PEARL-System Gebrauch gemacht.

3 Bechreibung der Schnittstellen

Die Schnittstelle des Prozeßrechners HP1000 besteht aus einer D/A-Karte mit 4 Analogausgängen á 8 Bit (negative Spannungen nicht möglich), 2 A/D-Karten mit je 12 Bit Analogeingängen und einer Binärschnittstelle (DAS) mit 16 Bit Ausgängen. Über die Binärschnittstelle wird der Hybridrechner gesteuert und das Vorzeichen der Stellgröße ausgegeben. Zusätzlich können auch Steuersignale für einen am Hybridrechner angeschlossenen X-Y-Schreiber übertragen werden.

4 Verwendung der Programmiersprache PEARL bei der Realisierung der Aufgaben auf der HP1000

4.1 Anforderungen an die Programmiersprache

Infolge der Aufgabenstellung, d.h. laufende Änderung der Regelalgorithmen, durfte die Programmiersprache keine aufwendige Programmarbeit erfordern, wie es bei Assemblersprachen der Fall ist.
Die Kommunikation mit dem Hybridrechner und eventuelle Änderungen der Ein- und Ausgangsbelegung sollte einfach sein. Die Echtzeitprogrammierung sollte ein nachvollziehbares (und berechenbares) Timescheduling haben, damit Schlüsse für die Anforderungen an ein später einzusetzendes Mikroprozessorsystem gezogen werden können.

4.2 Voraussetzung für die Benutzung der Sprache PEARL

Der Autor besaß Programmierkenntnisse in den Sprachen Pascal, FORTRAN (Großrechenanlagen) und der Assemblersprache für den Motorola 6802. Bezüglich einer Echtzeit-Prozeßsprache war hier echtes Neuland zu betreten. Vorteilhaft war, daß vom gleichen Institut auf ähnliche Weise vor diesem Projekt eine Simulation einer Regelung eines Blockheizkraftwerkes durchgeführt wurde. Die Erfahrungen, die hierbei mit PEARL und dem Prozeßrechner in Verbindung mit dem Hybridrechner gemacht wurden, konnten jedoch nur teilweise übernommen werden, da das Herstellerbetriebssystem der HP1000 vor Beginn dieses Projektes geändert wurde und der Betrieb sich darauf erst wieder stabilisieren mußte.

4.3 Vorteile und Nachteile von PEARL als Echtzeit-Programmiersprache

Die zur Verfügung stehende Literatur suggerierte die perfekte Erfüllung der Anforderungen an die Programmiersprache durch PEARL. Mängel stellten sich erst bei der Benutzung heraus.

4.3.1 Vorteile

Bestechend ist die einfache Handhabung der Sprachkonstrukte in PEARL, die sich bei der Programmierung positiv auswirkt. Bekannte Elemente von Hochsprachen (besonders von Pascal) mischen sich mit der logisch nachvollziehbaren prozeßeigenen Sprachwelt.

- Schnittstellenbehandlung
  Die Kommunikation der I/O-Schnittstellen ist einfach. Physikalische I/O-Änderungen werden nur im Systemteil des Programms durchgeführt. (Zeitersparnis und Gewinn an Übersicht).
- Programmaufbau
  Der strukturierte Programmaufbau und die fast unbegrenzte Namensgebung für Variable und Schnittstellen ist selbstdokumentierend und macht Programme auch nach längerer Zeit nachvollziehbar.
- Modultechnik
  Dieses Hilfsmittel ermöglicht die Einzelerprobung der Programme. Zusätzlich kann man mit wenigen Änderungen Programmteile für unterschiedliche Programmabläufe verwenden.
- Echtzeitsynchronisation
  Vernachlässigt man die Schnittstellenlaufzeit, so können Ereignisse auf einfache Art (fast) zeitgenau gesteuert oder registriert werden.
- Die Fehlermeldungen des Compilers sind ausreichend, man muß jedoch unnötige Warnungen von wichtigen unterscheiden lernen.
- Die Verwendung des Hersteller-Standardpaketes für Grafik durch Einbindung externer Routinen in PEARL stellte eine zwingend notwendige Erweiterung des Sprachumfanges dar.

4.3.2 Nachteile

Für den Benutzer suggeriert PEARL Großrechenkomfort. In Wirklichkeit ist PEARL auf einem Kleinrechner implementiert, dessen Betriebssystem den gewünschten Komfort nicht bieten kann.

- Sprachumfang
  Der Prozeßrechner wird im Einprozessor-Multitasking-

Betrieb gefahren, was bei einer Echtzeitprogrammierung zu erheblichen Nachteilen führt.Bei der Abarbeitung z.B. zweier gleichpriorisierter Prozesse muß der eine erst beendet sein,bevor der andere beginnen kann. Das kann die Zeitsynchronisation mit dem Hybridrechner zerstören (s.Bsp.Kaskadenregelung).
Für die Auslegung des Reglers ist eine Abschätzung der Rechenzeit zur Compile-Zeit notwendig, z.B. per Sprachkonstrukte. Es gibt bei PEARL im Gegensatz zu den Assemblersprachen nicht einmal eine Zeitangabe für die Dauer der Operationen je Herstellerimplementierung. Man muß sich mit der Abtastzeit regelrecht "herantasten". Änderungen der Regelalgorithmen können den gefundenen Rechenzeitwert wieder zunichte machen.

Warum gibt es bei PEARL keine Sprachelemente für die Grafik?
Die selbsterstellte Grafik sowie die Userroutine für die Rechnersystemzeit sind in anderen Sprachen realisiert und müssen beim Bindevorgang explizit angegeben werden.

- Laufzeit-Umgebung
  Verwirrend ist, daß es mehrere Ebenen gibt, aus denen Fehlermeldungen dem Benutzer erscheinen können.
  1) Fehlersignale werden vom PEARL-Laufzeitsystem angezeigt(magere Ausstattung).
  2) Fehlermeldungen können zusätzlich vom Hersteller-Betriebssystem kommen(Resourcenverwaltung).

  Ein Monitoring der PEARL-Laufzeit wird vermißt. Es wird nicht angezeigt, ob ein Prozeß soeben gelaufen ist und wie oft er im vergangenen Zeitraum gelaufen oder eingeplant ist.
  Unklarheiten bestanden bei der Analyse, warum Prozesse "abgestürzt" sind (mangelhafte Fehlermeldung).
- Programmentwicklungsumgebung
  Sehr Zeitaufwendig ist der Binde-Vorgang , besonders bei Verwendung von mehreren Moduln. Sehr nützlich wurde die Verwendung von Makros empfunden, die jedoch die gleiche Benennung der Objekt- und Bindefiles voraussetzt.
- Notwendig ist auch die Information des Zeitverlustes bei Taskwechsel. Laufen ineinander geschachtelte Programmteile mit unterschiedlichen Taktzeiten (s.Kaskadenregelung mit unterschiedlicher Abtastung für den Beschleunigungs- und Wegregelkreis), so darf der Taskwechsel die zeitgenaue Abarbeitung der Reglertasks nicht zerstören. Beim Multitasking-Betrieb hängt die Zeit des Taskwechsels noch zusätzlich von der Belastung des Rechners und der gesetzten Priorität der Task ab.

5 Ausblick

Der Autor hat sich bei seiner Problemlösung für die Methode der Echtzeit-Simulation entschieden. Hierbei wurde PEARL als Echtzeit-Simulationssprache eingesetzt. Sämtliche Problemstellungen können in PEARL vorteilhaft und zeitsparend implementiert werden.
Kritisiert wird, daß in PEARL keine Echtzeitzusicherungen vorhanden (Assertions) sind
Kritisiert wird, daß in PEARL keine Echtzeitzusicherungen (Assertions) vorhanden sind, mit denen Echtzeitanforderungen durch den Compiler überprüfbar wären.
Mangelnde Fehlermeldungen machten eine intensive Betreuung seitens der Spezialisten des Prozeßrechenzentrums notwendig. Besonders bei Laufzeit-Fehlersignalen ist eine stärkere Differenzierung der Fehlerquellen notwendig.
Eine Simulation zieht eine ausreichende Dokumentation nach sich. Deshalb ist es zwingend, daß die verwendete Simulationssprache Sprachkonstrukte für eine Grafik zur Verfügung stellt.

Literaturverzeichnis

Systematisches Programmieren mit PEARL
Herbert Brinkkötter,Klaus Nagel
Herbert Nebel und Klaus Rebensburg
Studien-Text, Ak.Verlagsges.Wiesbaden

Regelungstechnik I
Unbehauen
Vieweg Braunschweig/Wiesbaden

Abtastregelung
J.Ackermann
Springer-Verlag 1983