MEHRPROZESSOR-STEUERUNGSSYSTEM ZUR AUTOMATISIERUNG VON PRÜFSTÄNDEN

J. Schmidt[x]
D. Abbott[x]
B. Pothen[xx]
K. Zwoll[x]

[x] Kernforschungsanlage Jülich GmbH
- Zentrallabor für Elektronik -

[xx] Pierburg Luftfahrtgeräte Union GmbH, Neuß

1. August 1980

Zusammenfassung

Es wird ein Mehrprozessor-Steuerungssystem zur Automatisierung von Prüfständen vorgestellt. Das System zeichnet sich durch funktionsorientierten modularen Aufbau der Hardware und Software aus.

Entwicklungsschwerpunkte sind ein Echtzeit-Betriebssystem, das die Intertaskkommunikation in dem Mehrprozessorsystem steuert, eine Prüfsprache zur Programmierung des Prüfablaufes, sowie Softwarepakete mit digitalen Steuer- und Regelfunktionen.

1. Einleitung

Bei der Luftfahrtindustrie ist eine ausgefeilte Qualitätskontrolle und Sicherung seit Jahren schon wichtigster Teil der Komponentenfertigung. Deshalb unterliegen auch hydromechanische Kraftstoffregler für Flugzeugstrahltriebwerke vor der Auslieferung einer intensiven Prüfung. Hierzu dienen Prüfstände, auf denen nach einem genau festgelegten Prüfablauf Betriebszustände des Triebwerks simuliert und Prüfdaten ermittelt werden. Bisher werden die Prüfstände manuell bedient und die Prüfdaten handschriftlich in einem Prüfprotokoll dokumentiert. Diese Verfahrensweise ist ebenso kosten- wie zeitintensiv, deshalb bietet sich die Automatisierung der Prüfstände an.

2. Aufgabe und Funktion des Kraftstoffreglers

Hauptaufgabe des Kraftstoffreglers ist, den Flugkraftstoff mit Hilfe einer Hochleistungszahnradpumpe auf den vor der Brennkammer benötigten Druck zu befördern und den der Betriebsbedingung entsprechenden Kraftstoffffluß einzustellen /1/. Bei dem bisher manuell geführten Prüfablauf werden die Stellgrößen auf die durch die Prüfvorschrift gegebenen Werte eingestellt und die Meßgrößen abgelesen und protokolliert.

Die Steuerung des Prüfablaufes und die Protokollierung der Prüfdaten ist Aufgabe des im Aufbau befindlichen Automatisierungssystems.

3. Konfiguration des Mehrprozessor-Steuerungssystems zur Prüfstandautomatisierung

Bei der Konzeption des Mehrprozessor-Steuerungssystems wurde besonderer Wert auf einen modularen Aufbau von Hardware und Software gelegt, um eine flexible Anpassung auch an zukünftige Aufgaben und Systemerweiterungen zu erleichtern.

Das Prüfstandautomatisierungssystem (Abb. 1) besteht aus einem Zentralrechner (ZR) und drei prozeßnahen Prüfstandsteuereinheiten (PSSE). Der Zentralrechner versorgt über serielle Leitungen die Prüfstandsteuerungen mit Daten (Prüfablaufprogramme), archiviert die Prüfdaten und protokolliert Fehlermeldungen.

3.1 Aufbau und Funktion der Prüfstandsteuereinheit

Eine Prüfstand-Steuereinheit besteht aus vier Ein-Karten-Mikrorechnern (aktive Teilnehmer) und vier passiven Teilnehmern (Prozeßschnittstellen und Speicher), die über einen gemeinsamen Systembus verbunden sind (Abb. 2). Diese Konzept ermöglicht eine funktionsorientierte Aufgabenteilung auf einzelne Mikrorechner /2/.

Die Funktionen der Prüfstandsteuereinheit sind wie folgt auf die einzelnen Module verteilt.

Das Kernstück der Prüfstandsteuereinheit ist der Prüfablaufrechner. Aufgaben dieses Ein-Karten-Mikrorechners sind

- Initialisierung der Prüfstände beim Anfahren und Wiederanlauf nach Störung;
- Koordination der Aktivitäten aller Module;
- Interpretation des Prüfablaufprogrammes;
- Aufbereitung der Prüfdaten;
- Funktionsüberwachung (Fehlercheck) der Steuereinheit.

Ein zweiter Ein-Karten-Mikrorechner dient als Kommunikationssteuerwerk (Communication Controller). Er stellt die Verbindung zwischen der PSSE und dem Zentralrechner über eine serielle Datenleitung (20 mA-Stromschleife) her.

Seine Aufgaben sind:

- Generierung der untersten Protokollebene;
- Datensicherung, Übertragungsfehlererkennung (CRC) und Korrektur;
- Empfangen und Aufbereiten der Nachrichten des ZR (Steuerinformationen, Prüfablaufprogramme usw.);
- Übermittlung der Nachricht an den Prüfablaufrechner;
- Übernehmen der Nachrichten des Prüfablaufrechners (Prüfdaten, Ready-Meldung, Fehlermeldung) und Übertragung an den Zentralrechner.

Der dritte Mikrorechner arbeitet als digitaler Regler (DDC-Rechner) mit den Funktionen:

- Einstellen und Überwachen der Prüfparameter mit Hilfe digitaler Regelkreise durch
- P, PI oder PID-Algorithmen
- Bedienung von 12 Regelkreisen bei max. 10 Hz Abtastrate.

Um diese Abtastrate zu gewährleisten, werden für mathematische Operationen in den Regelalgorithmen mit einem schnellen Arithmetik-Prozessor durchgeführt.

Der Bedienfeldrechner mit angeschlossenem Bildschirm und Tastatur dient als Interface zwischen Bedienungsmann und Prüfstandsteuereinheit. Dazu ist erforderlich

- Verarbeitung der Bedienereingaben (über Tastenfeld) und deren Weitergabe an die Ablaufsteuerung sowie
- Bedienerführung durch Textausgaben über einen Bildschirmmonitor, digitale, quasi-analoge und graphische Meßwertanzeige (durch Balkendiagramme) über Bildschirm, ebenso optische und akustische Alarmmeldung.

Zwei Intelligente Analog Eingabe-Karten (IAE) sind, wie die DDC-Karte, mit einem Mikroprozessor und einem Arithmetik-Prozessor bestückt, der die Aufbereitung der Analog-Daten ermöglicht (z.B. digitale Glättung, Offset-Kompensation, Linearisierung von Meßwertaufnehmerkennlinien).

Die zweite Eingabe-Karte dient zur quasi-analogen Sollwerteingabe über Potentiometer bei Handbetrieb. Damit ist der manuelle Betrieb des Prüfstandes auch bei Ausfall des Zentralrechners möglich.

Zwei Analog-Output-Karten mit je 4 Ausgabekanälen bedienen die Stellglieder des Prüfstandes. Sie werden vom Systembus wie Speicherplätze angesprochen und setzen einen 12-bit-Binärwert in ein standardisiertes Analogsignal (4 - 20 mA-Strom) um.

Als Digitalinput/Output und Impulszählermodul dient ein weiterer Einkarten-Mikrorechner. Dieser steuert binäre Prüfstandfunktionen (Öffnen und Schließen von Ventilen, Überwachen von Ventil- und Schalterfunktionen), erfaßt Meßwerte, die als BCD-Zahlen vorliegen und zählt Impulse, die als Ausgangssignal eines Drehzahlmessers eintreffen.

Der Erweiterungsspeicher (C-MOS-RAM) ist mit einer Puffer-Batterie ausgestattet und schützt so den Speicher gegen Informationsverlust bei Netzausfall. Er dient in dem System zur vorübergehenden sicheren Speicherung wichtiger Betriebsinformationen und zum Informationsaustausch zwischen den aktiven Busteilnehmern (Ein-Karten-Mikrorechnern) im Sinne eines Briefkastenspeichers (Mailbox-RAM).

3.2 Hardware-Realisierung der Prüfstandsteuerung

Um die Gesamtentwicklungszeit zu minimieren, wird die Hardware soweit wie möglich aus OEM-Komponenten aufgebaut. Für die Prüfstandsteuerung wird das SIEMENS AMS85-System /3/ eingesetzt. Dieses Kartensystem kann wahlweise mit 8- oder 16-bit Wortbreite arbeiten und erlaubt einen direkt adressierbaren Speicherbereich von maximal 8M-Worten. Es wird das Doppel-Europa-Kartenformat verwendet. Die zur Zeit verfügbaren AMS-Ein-Karten-Mikrorechner /4/ bauen auf dem Mikroprozessor SAB 8085 auf. Sie sind ausgestattet mit karteneigenem RAM, EPROM, seriellen und parallelen Ein/Ausgabe-Schnittstellen sowie allen sonst notwendigen Hardware-Einrichtungen, die sowohl autonomen Betrieb (als Einzelrechner) als auch in einem Mehr-Prozessor-System-Bus erlauben. Ein AMS-Rechnertyp ist mit dem Arithmetik-Prozessor AMD 9511 erhältlich. Dies hat die Entscheidung für das AMS-System stark beeinflußt, da für die Digitale Regelung (DDC) eine schnelle Arithmetik Voraussetzung ist.

Der AMS-Bus ist die Doppel-Europa-Karten-Version des INTEL Multibus /5/. Beide Bus-Systeme sind in ihren Signalen kompatibel und können durch Adapter miteinander verbunden werden. Somit kann das AMS-System durch das große Angebot an Multibusplatinen ergänzt werden.

Die Hardware-Eigenentwicklung beschränkt sich auf die Intelligenten Analog-Eingabekarten und die Analog-Ausgaben. Sie sind, vom Systembus aus betrachtet, passive Teilnehmer (Slave), d.h. sie können selbst nicht auf den Systembus zugreifen, sondern müssen von einem aktiven Teilnehmer (Master) mit Daten versorgt werden.

Bei der Analog-Ausgabe-Karte wird das binäre 12-bit Ausgabedatum in ein Zwischenspeicherregister (Latch) wie in einen RAM-Speicherplatz geschrieben (Memory Mapped I/O). Über die Ausgangspins des Latches liegt es an den Digital-Analogwandlern solange an, bis ein neues Wort eingeschrieben wird oder über ein RESET-Signal der Analog-Ausgang hardwaremäßig auf Null gesetzt wird.

Die Kommunikation zwischen einem Master und der Intelligenten Analog-Eingabe verläuft über ein Dual-Port-RAM auf der Eingabe-Karte. In festgelegten RAM-Bereichen werden die Informationen über Datenblöcke - Steuerparameter und Meßwerte - ausgetauscht.

Die beiden selbstentwickelten Module wurden zunächst als Prototyp auf Universal-Interface-Karten in Wire-Wrap-Technik aufgebaut und sollen nach eingehender Erprobung als gedruckte Schaltung hergestellt werden.

Weitere selbstentwickelte Karten dienen zur galvanischen Trennung von Prüfstand und Steuereinheit. Sie haben jedoch keine prinzipielle Bedeutung für das Mehrprozessor-Steuerungssystem und werden deshalb im folgenden nicht behandelt.

3.4 Aufbau und Funktion des Zentralrechners

Der Zentralrechner dient zur Steuerung und Überwachung des gesamten Prüfstand-Automatisierungssystems. Das Blockschaltbild in Abb. 3 zeigt den modularen Aufbau.

Der Ein-Karten-Mikrorechner arbeitet als System-Master unter einem Betriebssystem, das aus dem Realtime-Multitasking Betriebssystem INTEL RMX80 /6/ weiterentwickelt ist. Er dient zum

- Dialog mit dem Bedienpersonal über das Datensichtgerät;
- Erstellung und Speicherung der Prüfablaufprogramme;
- Störungsmeldung und Protokollierung.

Der Floppy-Disk-Controller steuert zwei Double-Density-Laufwerke, auf denen Betriebssystemdateien, Prüfablaufprogramme und Prüfprotokolle gespeichert sind.

Der Kommunikations-Controller steuert mit Hilfe eines erweiterten RMX80-Betriebssystems den Dialog zwischen der Zentrale und den Prüfständen (vgl. auch Kap. 3.1).

Der Zentralrechner ist aus OEM-Modulen der Serie INTEL iSBC80 aufgebaut.

In Abb. 3 ist weiterhin die Interface-Baugruppe zu sehen, über die die Prüfstände in Zukunft in ein globales Produktionssteuerungssystem integriert werden sollen.

4. Entwicklungsschwerpunkte

Die Hardware des Prüfstandautomatisierungssystems konnte weitgehend aus OEM-Komponenten aufgebaut werden. Die eigenen Hardware-Entwicklungen konzentrieren sich daher auf die Prozeßinterfacekarten, die bereits besprochen wurden.

Schwerpunkte der Software-Entwicklung sind die Betriebssoftware für das Mehrprozessorsystem, die Kommunikationssoftware mit Protokollerstellung und Fehlererkennung, die Prüfablaufsteuerung mit einer Prüfsprache sowie ein umfangreiches Paket von prozeßorientierten Steuer- und Regelfunktionen.

4.1 Mehrprozessorkommunikation

Multi-Mikrorechnersysteme lassen sich nach der Art ihrer Module in fest und lose gekoppelte Systeme einteilen /7/.

Bei fest gekoppelten Systemen läuft die Datenübertragung parallel über ein gemeinsames RAM oder FIFO. Sie wird für zeitkritische Anwendungen eingesetzt, d.h., wo kurze Übertragungs- und Reaktionszeiten notwendig sind, und die gekoppelten Teilnehmer räumlich nahe beieinander liegen. Es entfällt hier in der Regel eine besondere Vorkehrung gegen Übertragungsfehler.

Größere Entfernungen werden aus Kostengründen mit serieller Datenübertragung in loser Kopplung überbrückt. Da Fehler bei der Datenübertragung hierbei nicht mehr auszuschließen sind, ist ein geeignetes Protokoll mit entsprechender Datensicherung erforderlich.

Im vorliegenden Projekt werden beide Kommunikationsmethoden eingesetzt: Feste Kopplung über die Systembusse innerhalb der Prüfstand-Steuereinheit bzw. des Zentralrechners, die miteinander über Communications-Controller und eine serielle Leitung lose gekoppelt sind.

4.2 Mehrprozessorbetrieb in der Prüfstand-Steuereinheit

Zum Betrieb mehrerer aktiver Teilnehmer in fester Kopplung an einem gemeinsamen Systembus sind Hardware- und Software-Einrichtungen nötig, die den Datenverkehr (Buszugriff und Datenaustausch) über den Bus sichern und koordinieren.

Auf jeder Masterkarte der AMS85-Baureihe befindet sich eine Hardware-Logik, die den Buszugriff steuert. Durch geometrische Prioritätszuteilung (Daisy Chain) wird die Priorität eines jeden aktiven Teilnehmers im System bestimmt. Bei einem Buszugriff werden den Karten niederer Priorität durch ein Sperrsignal (BPRO/=Bus Priority Out/) untersagt, den Bus zu belegen. Während des Buszugriffes wird auf das Bus-Belegt-Signal (BUSY/) gesetzt, womit auch höherpriore Master daran gehindert werden, einen laufenden Transfer zu unterbrechen. Nach Beendigung des Buszugriffes setzt der Master das Sperr- und das Belegt-Signal wieder zurück und gibt damit den Bus für die übrigen Teilnehmer frei. Die Daisy Chain eignet sich wegen der Gatterdurchlaufzeiten nur für AMS-Systeme mit maximal drei Master-Karten. Werden, wie im hier vorgestellten System, mehr als drei Master eingesetzt, muß eine externe Priorität-Decoder-Logik auf einer zusätzlichen Karte aufgebaut werden. In dem AMS-Bussystem kommunizieren aktive Busteilnehmer über ein gemeinsames RAM (Mailbox).

Durch einen Synchronisationsmechanismus, sogenannte "Semaphoren" /2, 8/, wird eine korrekte Datenübermittlung vom sendenden u-Rechner 'A' zum empfangenden u-Rechner 'B' garantiert.

Semaphoren sind im vorliegenden Anwendungsfall Signalbytes in Speichern am gemeinsamen Systembus. Sie steuern den Zugriff zu einem definierten RAM-Bereich (Mailbox), über den Nachrichten ausgetauscht werden.

Um eine störungsfreie Kommunikation zu garantieren, muß auch der gleichzeitige Zugriff auf das Semaphorenbyte ausgeschlossen werden, da sonst mehrere Busteilnehmer die Semaphore als "frei" vorfinden und versuchen könnten, sie zu sperren und auf die Mailbox zuzugreifen.

Das AMS85-System bietet eine Hardwarelösung für den notwendigen gegenseitigen Ausschluß (mutual exclusion) des Buszugriffes: Die BUS-LOCK-Funktion.

Ein aktiver Busteilnehmer kann den Systembus für sich reservieren, indem er über ein Flip-Flop die Bus-Belegt-Leitung (BUSY/) aktiv schaltet und solange festhält, bis er die Semaphorenoperation abgeschlossen hat. Sein Zugriff über den Bus auf das RAM kann nicht durch höher priore Teilnehmer unterbrochen werden.

Das Flußdiagramm in Abb. 4 zeigt den Ablauf einer Semaphoren-Operation.

Die Synchronisation des Mehrfachzugriffes durch Semaphoren ist eine einfache und flexible Methode der Mehrprozessor-Kommunikation. Der Rechenzeitbedarf für die Semaphorenverwaltung ist relativ gering, so daß die Methode für Realtime-anwendung eingesetzt werden kann.

Die Semaphore ist das Steuerelement in der untersten Ebene der Betriebssoftware für die Mehrprozessorkommunikation.

4.3 Höhere Betriebssystemebenen

Zur Steuerung der Datenströme in einer komplexen Anlage wie dem vorgestellten Mehrprozessorsystem muß dem Anwendungsprogrammierer ein geeignetes Betriebssystem zur Verfügung stehen.

Mehrprozessor-Betriebssysteme wurden im Hochschul- und Forschungsbereich entwickelt /9, 10/ und einige industrielle Pilotimplementationen sind bekannt /11, 12/. Für die verwendeten OEM-Mikrorechner ist zur Zeit kein Multi-Prozessor-Betriebssystem von den Herstellern erhältlich. Deshalb wurde das für die Mikrorechner der Serien INTEL iSBC80 erhältliche Realtime-Multitasking Betriebssystem RMX80 für Mehrprozessoranwendungen erweitert und durch eine höhere, bedienerorientierte Kommandoebene ergänzt.

RMX80 (Realtime Multitasking Executive) ist ein Software-Bausteinsystem, mit dem Betriebssystemfunktionen zur Intertask-Kommunikation und Synchronisation implementiert werden können. Da RMX80 für kleinere Systeme konzipiert ist, sind die Software-Schnittstellen zu den einzelnen Funktionen verhältnismäßig einfach und erfordern vom Anwender die Berücksichtigung vieler systemspezifischer Details.

In größeren Systemen, insbesondere bei Mehrprozessoranwendungen, File handling usw., sollte der Anwendungsprogrammierer nicht mit den Einzelheiten der Intertask-Synchronisation belastet werden, wenn er eine simple Ein/Ausgabe durch führt. Außerdem sollte das System einheitliche Schnittstellen für alle I/OS vorsehen, unabhängig davon, ob es sich um eine reale Datenendstation, wie eine Magnetplatte, oder eine Task auf demselben oder einem anderen Prozessor handelt. Das heißt: Ein/Ausgaben müssen geräteunabhängig sein. Andererseits muß man sich der Einschränkungen von Mikrorechnersystemen bewußt sein und darf nicht versuchen, einen voll ausgestatteten Minirechner aufzubauen.

Das Konzept der Geräteunabhängigkeit wird durch "logische Kanäle" implementiert. Eine Task, die eine Ein/Ausgabe durchführen oder mit einer anderen Task Daten austauschen möchte, öffnet (OPEN) zunächst einen logischen Kanal zu dem Kommunikationspartner. Der Datentransfer wird mit READ- bzw. WRITE-Aufrufen gesteuert. Nach Abschluß der Datenübertragung wird der logische Kanal wieder geschlossen (CLOSE) und steht dann weiteren Tasks zur Verfügung.

Jede Datenquelle oder -senke in dem System wird durch einen Treiber (Device-Driver) representiert. Dies ist ein Satz von Tasks, die die spezifischen Eigenarten der "Devices" in die einheitlichen Schnittstellen mit den Aufrufen OPEN, READ, WRITE und CLOSE umsetzen. Der Treiber wird durch seinen Namen (2 ASCII-Zeichen), der in einer Tabelle (Device Table) steht, aufgerufen. Für zusätzliche Geräte braucht nur ein neuer Treiber geschrieben und sein Name in die Tabelle eingetragen zu werden. Über die Treiber-Ebene synchronisieren die RMX80 Funktionen die Intertask-Kommunikation.

Ein Beispiel für einen geräteunabhängigen I/O ist die Kommunikation zwischen dem Zentralrechner und den Prüfstandsteuereinheiten. In Abb. 5 ist gezeigt, wie eine Anwendertask "RECORD" Daten von einer Task "DATACQ" in der Ablaufsteuerung liest.

Zunächst müssen beide Tasks durch OPEN-Aufrufe die Verbindung zu dem jeweiligen Kommunikationspartner herstellen. Für RECORD hieße das:

```
CALL  OPEN(.CHNL1,.(':LØ:DATACQ.ABL',Ø),5Ø,5Ø,.STATUS);
```

Darin ist CHNL1 eine Variable, in die die logische Kanalnummer eingetragen wird. LØ ist der Name der seriellen Datenleitung Nr. Ø, ABL identifiziert den Prozessor in der Prüfstandsteuereinheit, auf dem die Task DATACQ läuft, hier die Prüfablaufsteuerung. STATUS ist eine Variable für Fehlermeldungen. Die Zahlen 5Ø sind Timeout-Parameter, die aber hier nicht weiter behandelt werden sollen.

Das Statement bewirkt einen OPEN-Aufruf an den Treiber des Gerätes LØ. In diesem Fall gibt der Treiber den Aufruf an den Kommunikations-Controller, der seinerseits über die serielle Leitung ein OPEN an das Kommunikations-Modul der Prüfstandsteuereinheit schickt.

Inzwischen hat DATACQ aufgerufen

```
CALL OPEN(.CHNL1,.(':LØ:RECORD.SUP',Ø),5Ø,5Ø,.STATUS);
```

Dieser Call geht zu dem Kommunikationsmodul, das die beiden Aufrufe vergleicht und, wenn sie zueinander gehören, Bestätigungen an die anfordernden Tasks schickt.

Nachdem die Verbindung hergestellt ist, kann der Datentransfer beginnen. Die Task RECORD liest mit

```
CALL READ(CHNL1,.BUFFER,8Ø,.ACTUAL,.STATUS);
```

8O Bytes über den logischen Kanal CHNL1, beginnend mit der RAM-Position BUFFER. Die gelesene Anzahl von Bytes wird in ACTUAL eingetragen, Fehlermeldung in STATUS abgesetzt.

Entsprechend muß die Task DATACQ schreiben mit

```
CALL WRITE(CHNL1,.DATABUFF,COUNT,.STATUS);
```

DATACQ meldet damit, daß COUNT Bytes, beginnend mit der Adresse DATABUFF, an den logischen Kanal 1 geschickt werden sollen. COUNT muß kleiner oder gleich dem Zählparameter in dem korrespondierenden READ-CALL sein.

Nachdem die Datenübertragung von DATACQ nach RECORD beendet ist, schließen beide Tasks mit

```
CALL CLOSE(CHNL1,.STATUS);
```

den logischen Kanal.

4.4 Prüfsprache und Prüfablaufsteuerung

Die Entwicklungsbereiche "Prüfsprache" und "Prüfablaufsteuerung" werden zusammenhängend betrachtet, da sie funktionell eng miteinander verknüpft sind. Die Systemkonfiguration aus Abb. 1 ist in Abb. 6 noch einmal explizit für einen einzelnen Prüfstand dargestellt.

Für den Zentralrechner erscheinen Prüfstand, Prüfstandsteuerung und Bediener als eine Funktionseinheit - eine virtuelle Maschine.

Der Prüfablauf ist in einem Prüfprogramm festgelegt. Hierfür wird eine einfache Prüfsprache entwickelt, da die existierenden Prüfsprachen, wie z.B. ATLAS /13, 14/, für diese Anwendungen zu umfangreich sind und mit zur Verfügung stehenden Mitteln nicht implementiert werden können.

Der Quellcode des Prüfablaufprogrammes wird auf dem Zentralrechner erstellt und gespeichert. Ein Übersetzungsprogramm "PRAS" (Prüfablaufsprache) wandelt den Quellcode in den Maschinencode der virtuellen Maschine um. Er enthält die Prüfsätze und Bedienerinformationen, zusammengefaßt zu Prüfschritten. Das Prüfablaufprogramm wird für jeweils einen Prüfschritt über die serielle Datenleitung an die Prüfablaufsteuerung geschickt.

Der Quellcode der Prüfablaufsprache ist assemblerähnlich aufgebaut. Der virtuelle Maschinencode ist lediglich eine komprimierte Version des Quellcodes, in dem das erste Byte jedes Statements den Befehlscode enthält, alle weiteren Bytes sind Parameter oder Daten. Ein Interpreter benutzt jeweils das Befehlsbyte als Schlüssel für die Unterprogrammabrufe. Diese starten die Maschinenaktivitäten, indem sie mit Hilfe der Parameter die Daten an die entsprechende Funktionseinheit übergeben, z.B. Ventilstellung oder Sollwert eines Regelkreises.

Abb. 7 zeigt einige Statements eines Prüfprogrammes.

4.5 DDC-Softwarepaket

Bei der Entwicklung des DDC-Softwarepaketes wurde pragmatisch vorgegangen: Aus zahlreichen Veröffentlichungen über praktische Implementationen digitaler Regelungssysteme geht hervor, daß klassische PID-Algorithmen auch heute noch sehr weit verbreitet sind /15/. Man findet dafür mehrere Gründe:

1. Technikern und Anlagenbedienern ist die PID-Regelung aus der analogen Regelungstechnik vertraut.

2. Umfangreiche praxisbezogene Literatur bietet viele nützliche Entwurfs- und Optimierungshilfen /16, 17/ und verkürzt damit die Entwicklungszeit.

3. Die Vorteile vieler höherer Algorithmen lassen sich oft nur unter idealisierten Bedingungen nutzen, z.B. exakte Kenntnis der Regelstreckeneigenschaften.

Aus diesen Gründen wird in dem DDC-Mikrorechner ein PID-Algorithmus eingesetzt. Der Algorithmus kann an die Eigenarten der zu regelnden Anlage angepaßt werden.

Es können:

- P, PI oder PID-Algorithmen eingesetzt werden, abhängig von den verwendeten Stellgliedern als
- Stellungsalgorithmen (z.B. mit Analog-Digital-Umsetzern) oder als
- Geschwindigkeitsalgorithmen (mit Stellmotoren).

Das DDC-Softwarepaket enthält Routinen zur On-line-Parameterbestimmung und -Optimierung. Damit kann dem Regelkreis ein gewünschtes Verhalten (Sprungantwort mit definiertem Einschwingen) eingeprägt werden /17/.

Das Programm ist für 12 Regelkreise mit einer Abtastrate von 10 Hz ausgelegt, die bei weiterer Optimierung noch gesteigert werden kann.

5. Zukünftige Weiterentwicklung der Prüfstandautomatisierung

Das Prüfstand-Automatisierungssystem in dem hier besprochenen Ausbau stellt eine Zwischenstufe der Entwicklung dar. In Zukunft ist geplant, weitere Prüfstände anzuschließen und das Gesamtsystem in das im Aufbau befindliche Produktionssteuerungssystem zu integrieren /18, 19/. Damit ergibt sich eine On-line Kopplung zum implementierten Betriebsdatensystem und zum Betriebsrechner IBM 370.

Literatur:

1. K. Zwoll, V. Hendrix, B. Pothen, J. Schmidt:
Rechnergeführte Automatisierung von Regelgeräte-Prüfsystemen.
Spezielle Berichte der Kernforschungsanlage Jülich, JÜL-Spez. 51, 1979.

2. G. Adams, T. Rolander:
Design Motivations for Multiple Processor Microcomputer Systems.
Computer Design, März 1978, S. 81-89.

3. SIEMENS: AMS85 Bus System.
Technical Description, Febr. 1979.

4. SIEMENS: AMS85-D2/D3 Zentralcomputer.
Technische Beschreibung, Dez. 1979.

5. INTEL: iSBC80 Multibus.
INTEL-Publication 98-638, 1977.

6. INTEL: RMX80 User's Guide.
INTEL-Publication 9800522 B, 1978.

7. M.G. Gable:
Communications in Distributed Systems (Part I: Interfacing Techniques).
Computer Design, Febr. 1980, S. 30-34.

8. E.W. Dijkstra:
Solution of a Problem in Concurrent Programming Control.
Communications of the ACM Vol. 8, Nr. 9, Sept. 1965.

9. I. Jurca:
A Multiprocessor System with Multitasking Facilities.
Dissertation Technische Hogeschool Delft, 1977.

10. K. Meißner:
Ein homogenes, modulares Multi-Mikroprozessorsystem für Realtime-anwendungen.
Dissertation Ruhr-Universität Bochum, 1978.

11. K. Kuhn:
Mehrrechner- und echte Multiprozessor-Systeme mit fertigen uP-Platinen.
Elektronik 1979, Heft 8, S. 77-80.

12. Autorenkollektiv:
Modulares Mehrprozessor-Steuersystem (MPST).
Systembeschreibung 9.79; Hrsg. MPST-Arbeitskreis, GFK Karlsruhe, Projekt PFT.

13. D.B. Loveman:
Language Design for Automatic Test Equipment.
Conference Record AUTOTESTCON 1977, 2.-4. Nov. 1977, Hyannis, MA, USA.

14. I.G. Easton:
Electro Conference Record, 1977,
19.-20. April 1977, New York, USA.

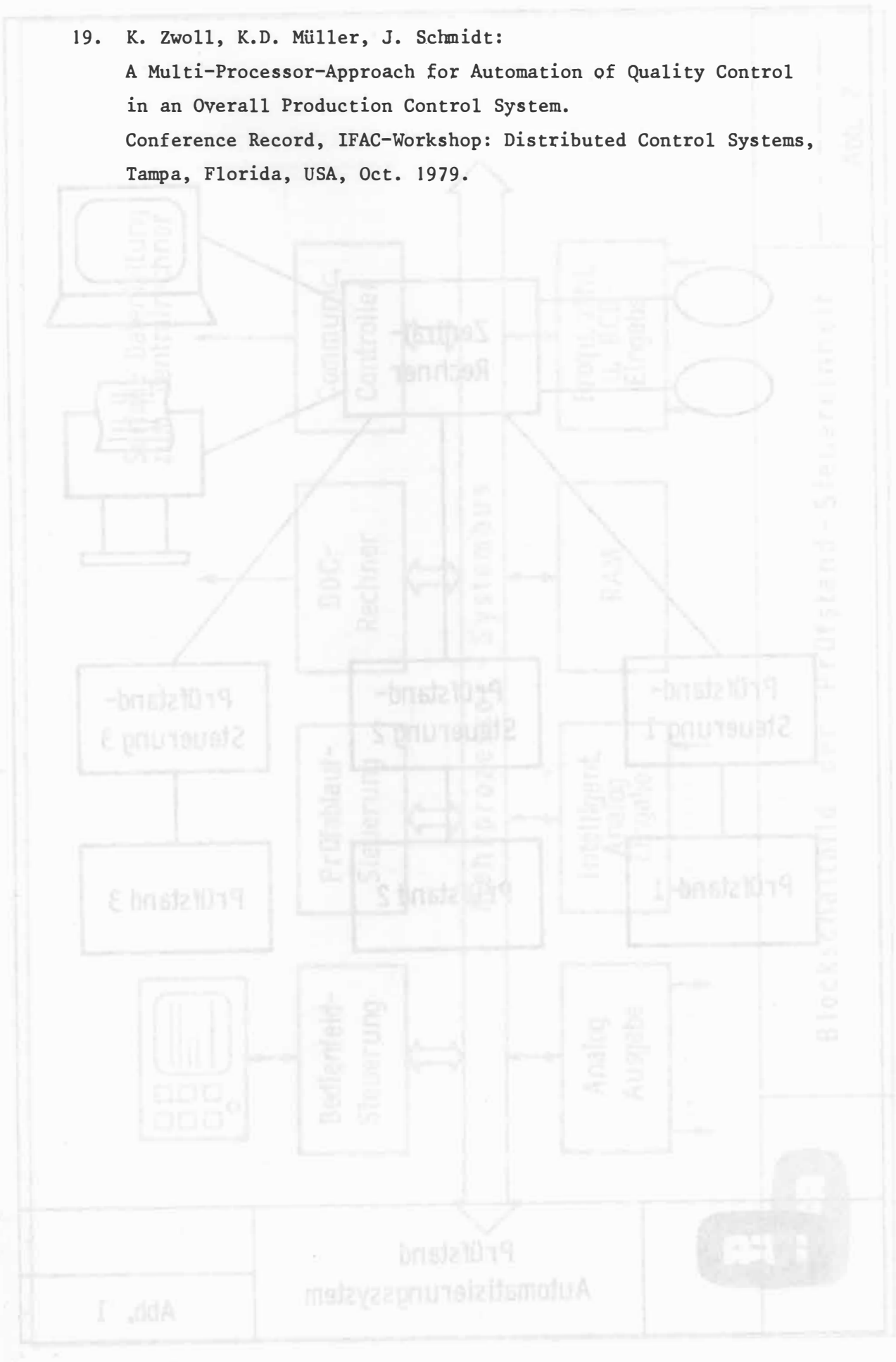
15. W. Latzel:
Regelung mit dem Prozeßrechner (DDC). Wissenschaftsverlag-Bibliographisches Institut, Zürich 1977.

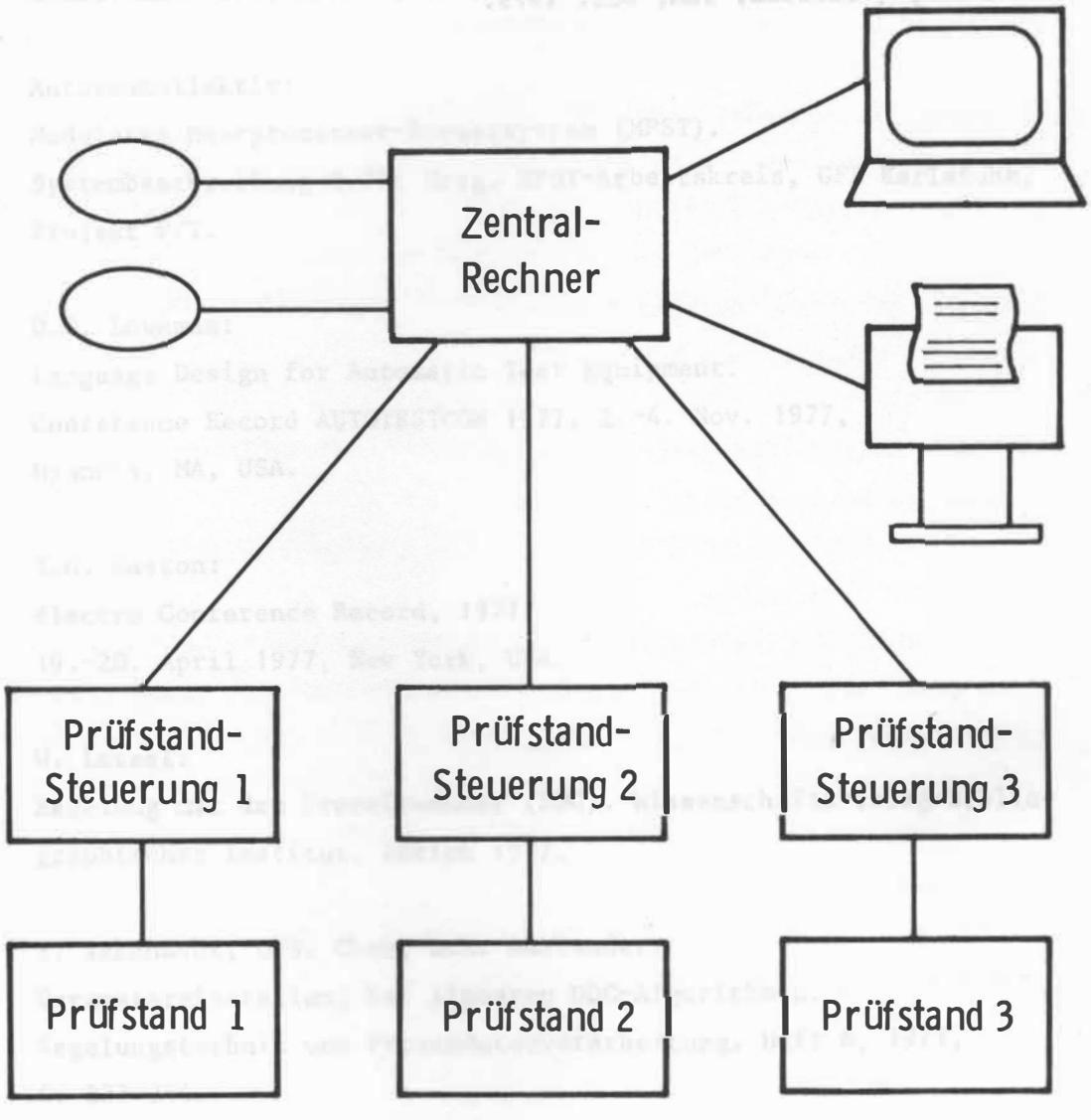
16. Y. Takahashi, C.S. Chan, D.M. Auslander:
Parametereinstellung bei linearen DDC-Algorithmen.
Regelungstechnik und Prozeßdatenverarbeitung, Heft 6, 1971,
S. 237-244.

17. SIEMENS-Applikation:
SMP-FSL-Demonstrationsmodell für die Anwendungen Messen, Steuern und Regeln.
Ausg. 5.80.

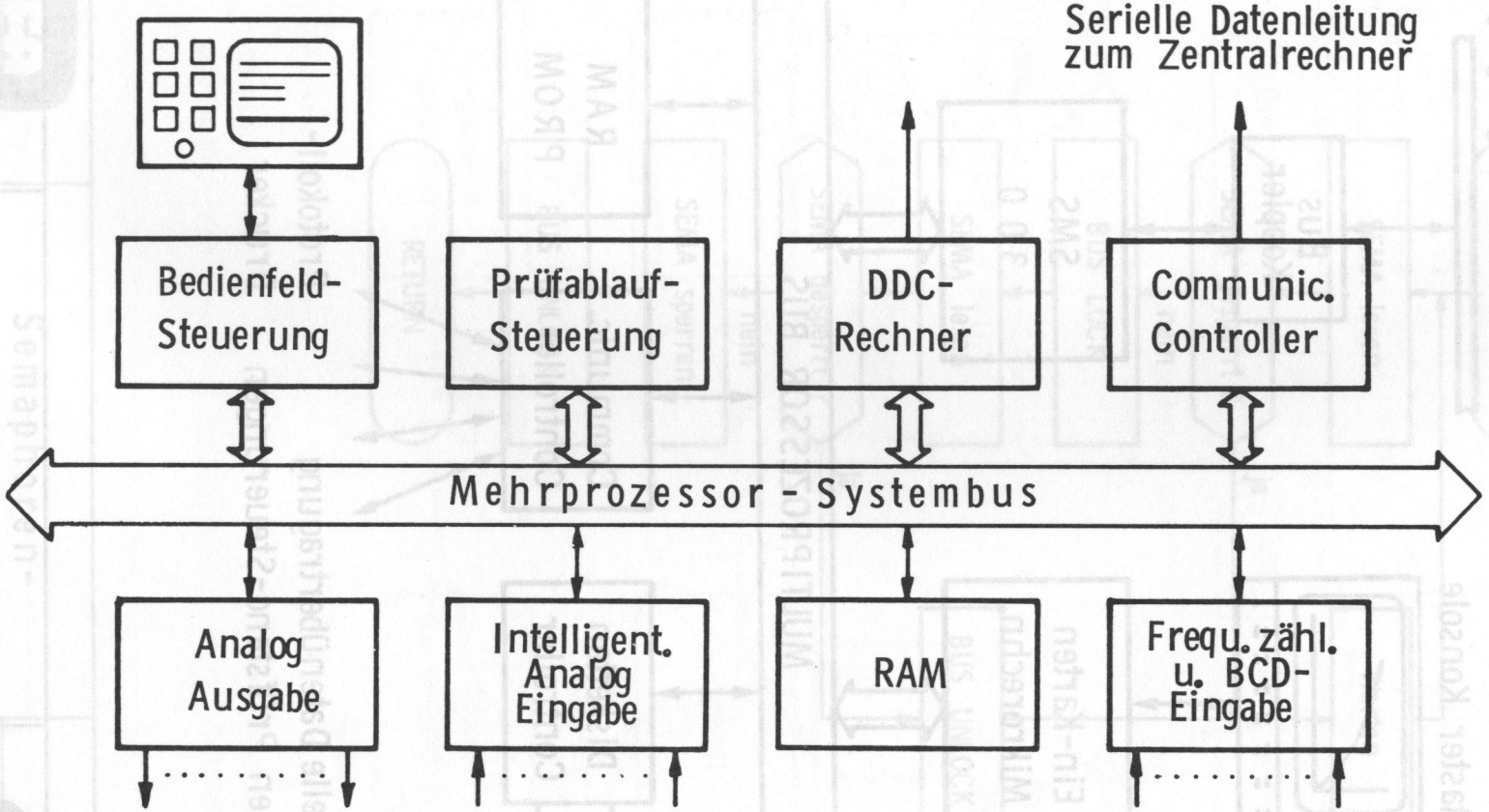
18. F. Billen, K. Zwoll:
Ein hochstandardisiertes Prozeßsteuerungssystem zur rechnergestützten Produktionssteuerung einer Kraftstoffreglerfertigung.
Realtime-Data-Handling and Process Control, H. Meyer, North-Holland Publ. Com., Brüssel and Luxemburg, 1980.

19. K. Zwoll, K.D. Müller, J. Schmidt:
A Multi-Processor-Approach for Automation of Quality Control in an Overall Production Control System.
Conference Record, IFAC-Workshop: Distributed Control Systems, Tampa, Florida, USA, Oct. 1979.





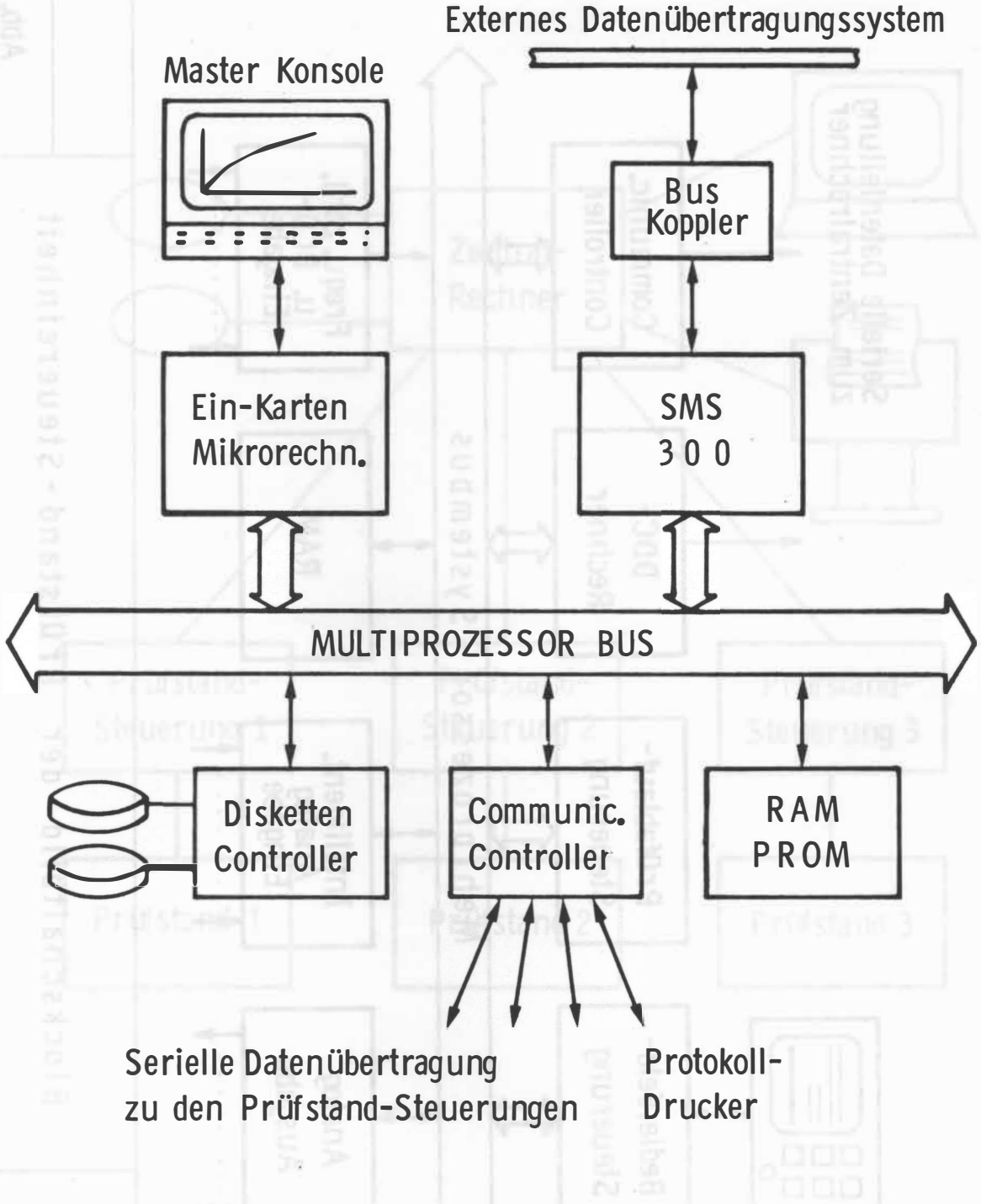
KFA	Prüfstand Automatisierungssystem	Abb. 1



KFA

Blockschaltbild der Prüfstand-Steuereinheit

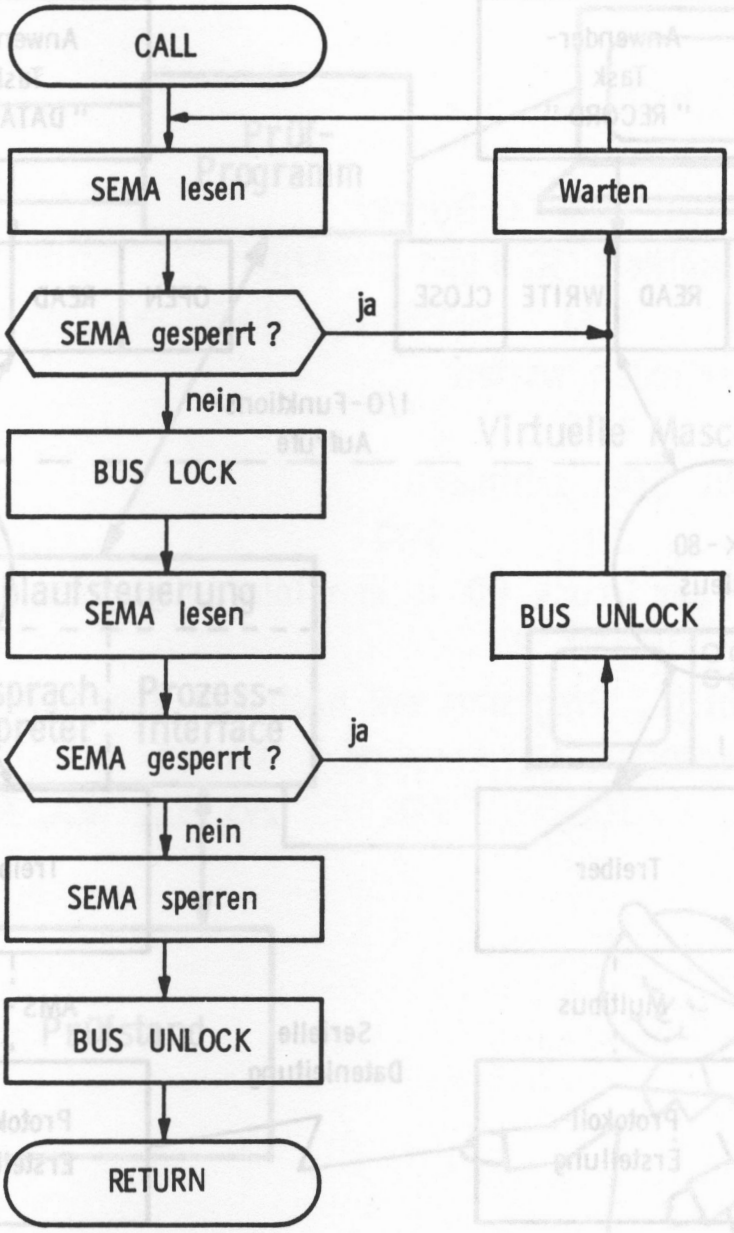
Abb. 2



KFA

Blockschaltbild
des Zentralrechners

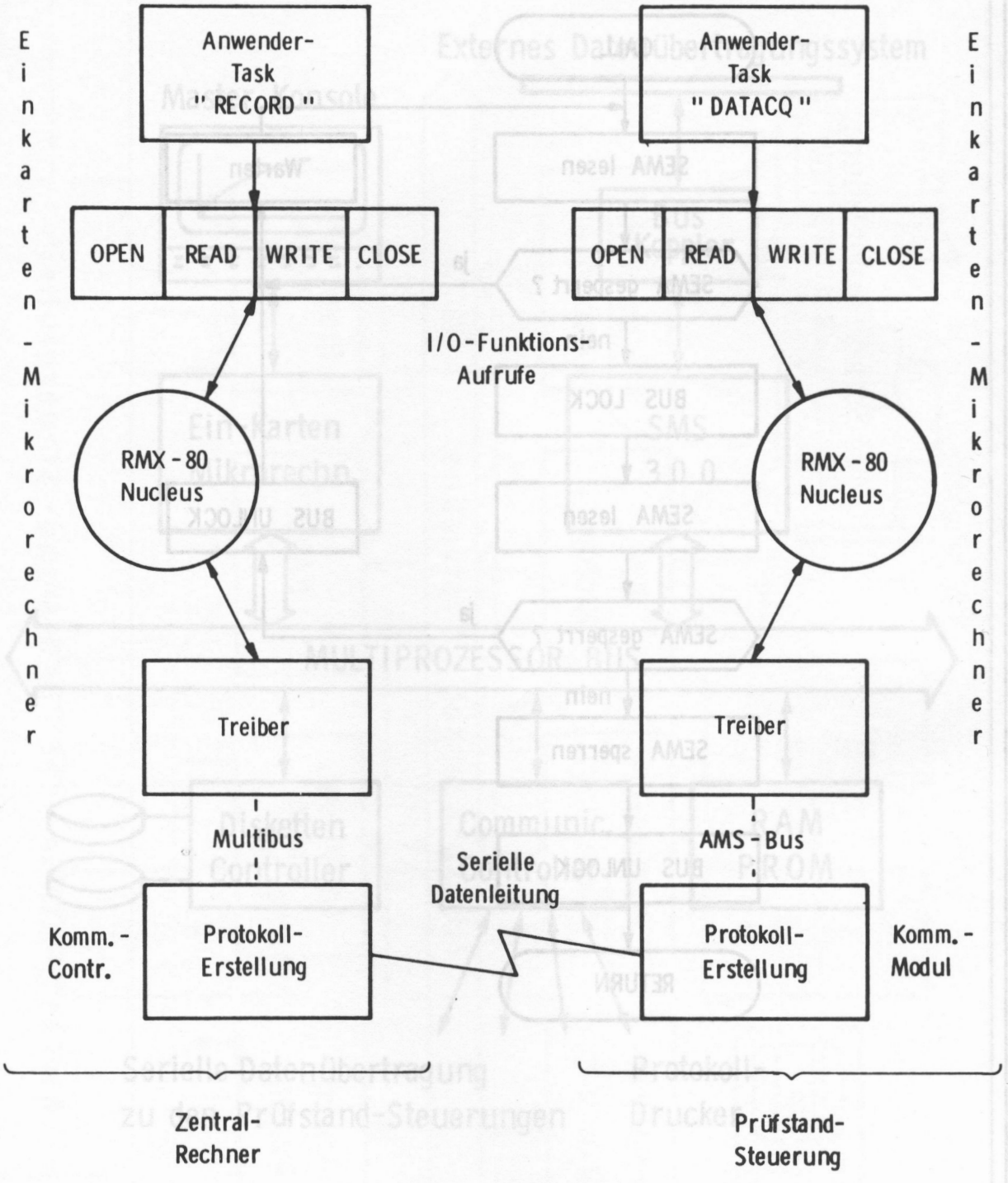
Abb. 3



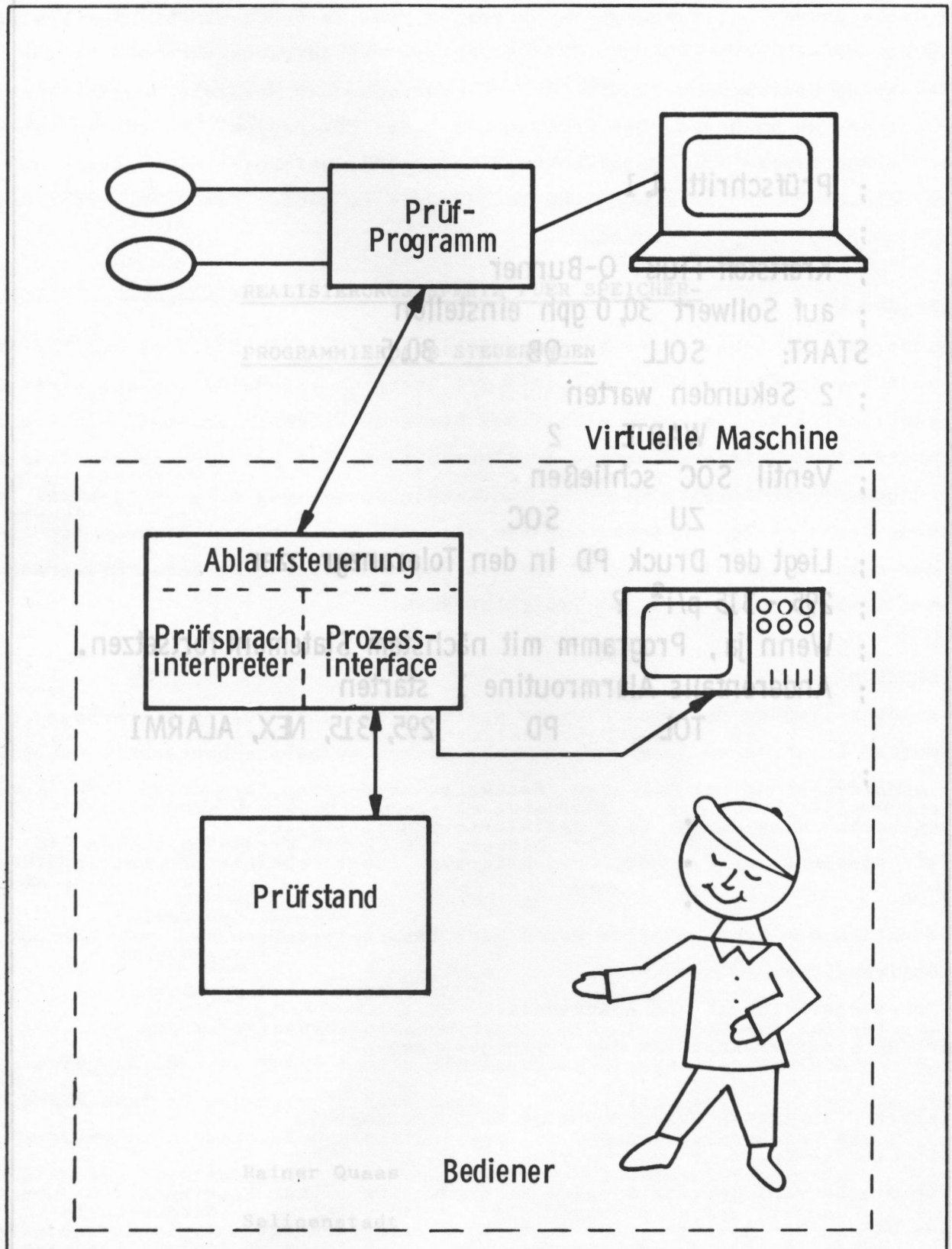
KFA

Semaphoren-Operation

Abb. 4



KFA	Mehrprozessor - Kommunikation über logische Kanäle	Abb. 5

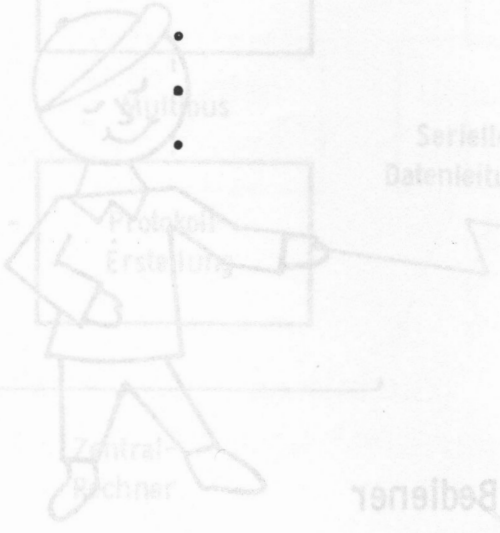


KFA

Steuerung der virtuellen Maschine

Abb. 6

```
;  Prüfschritt  4.1
;
;  Kraftstoff-Fluß  Q-Burner
;  auf Sollwert  30,0 gph  einstellen
START:      SOLL      QB        30.5
;  2  Sekunden  warten
            WARTE     2
;  Ventil  SOC  schließen
            ZU        SOC
;  Liegt der Druck  PD  in den Toleranzgrenzen
;  295 - 315 p/i²  ?
;  Wenn ja,  Programm mit nächstem Statement fortsetzen.
;  Anderenfalls Alarmroutine 1  starten
            TOL       PD        295, 315, NEX, ALARM1
;
                .
                .
                .
```



KFA	Source - Code eines Prüfprogrammes	Abb. 7