KERNFORSCHUNGSZENTRUM

KARLSRUHE

März 1976 KFK 2270

Institut für Datenverarbeitung in der Technik

MECROS

Ein System zur Messung von Leistungskenngrößen des Realzeitbetriebssystems CALAS 70

H. Grauer, M. Didic

GESELLSCHAFT
FÜR
KERNFORSCHUNG M.B.H.

KARLSRUHE

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.

KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

KFK-2270

Institut für Datenverarbeitung in der Technik

MECROS

Ein System zur Messung von Leistungskenngrößen des Realzeitbetriebssystems CALAS70

H. Grauer
M. Didic

Gesellschaft für Kernforschung m.b.H., Karlsruhe

Kurzfassung

Um das Leistungsverhalten von komplexen Realzeitbetriebssystemen analysieren zu können, sind Daten erforderlich, die nur aus dem im Betrieb befindlichen System gewonnen werden können. Softwaremeßverfahren stellen eine flexible und effektive Methode dar, mit der man Zugang zu den relevanten Daten eines Betriebssystems erhält. Dieser Bericht beschreibt die Grundlagen von Softwaremeßverfahren und ein experimentelles Softwaremeßsystem zur Messung der Leistungskenngrößen des Realzeitbetriebssystems CALAS7O.

MECROS - A system for the measurement of perfomance characteristics of the real-time operating system CALAS7O

Abstract

To analyze the performance of complex real-time operating system data are required, that can be obtained only from inside the running system. Software measurement techniques are a flexible and effective tool to get access to all relevant data of an operating system. This paper describes the principles of software measurement systems and an experimental software measurement system for the measurement of performance characteristics of the real-time operating system CALAS7O.

Inhalt

1. Einleitung

Die Messung und Analyse von Leistungskenngrößen eines Betriebssystems ist eine wesentliche Voraussetzung für die Beurteilung der Güte dieses Betriebssystems sowie für die Planung von Erweiterungen der Anlage und der Betriebssoftware. Realzeitbetriebssysteme sind dadurch gekennzeichnet, daß bestimmte Antwortzeiten auf äußere Anforderungen eingehalten werden müssen. Sie reagieren deswegen besonders empfindlich auf Änderungen in ihrem Aufgabenprofil. Im Grenzfall kann schon eine geringfügige Änderung des Aufgabenprofils durch Hinzufügen einer neuen Aufgabe zu einem Zusammenbruch des gesamten Systems führen. Daher sind gerade bei Realzeitbetriebssystemen Daten über die Systemauslastung und die Ausnutzung der Betriebsmittel von großer Bedeutung.
Meßverfahren zur Messung des Leistungsverhaltens von Betriebssystemen sind bereits bei mehreren großen Time-Sharing-Systemen erprobt worden /2/.

Die Entwicklung des experimentellen Meßsystems MECROS (MEASUREMENT and EVALUATION of the CHARACTERISTICS of a REAL-TIME OPERATING SYSTEM) sollte dazu dienen Erfahrungen beim Entwurf und beim Einsatz eines Softwaremeßsystems für ein Realzeit-Betriebssystem auf einem mittleren Rechner mit einfacher Hardwarestruktur zu sammeln und Aussagen über das Verhalten des Betriebssystems unter verschiedenen Aufgabenprofilen zu gewinnen.

Das dazu messende Betriebssystem ist das Realzeit-Betriebssystem CALAS7O, welches am Institut für Datenverarbeitung in der Technik der Gesellschaft für Kernforschung entwickelt wurde. CALAS7O wurde auf dem Telefunkenrechner TR86 implementiert. CALAS7O wird zur Zeit im Laborautomationssystem NICOLE am Hochflußreaktor in Grenoble bei der Kontrolle von fünf kernphysikalischen Experimenten eingesetzt und diente als Grundlage für das Telefunken-Realzeit-Betriebssystem RESY7O.
CALAS7O ist ein interaktives Realzeit-Betriebssystem mit Mehrfachzugriff und File-System.

Es ist nach dem Task-Konzept aufgebaut /4, 5/. Den Kern des Betriebssystems bildet der Scheduler mit der zugehörigen Auftragsliste. Die Auftragsliste ist in Ebenen mit aufsteigenden Softwareprioritäten geordnet. Jedes Programm besitzt eine Ausführungspriorität, die ein Maß für seine Wichtigkeit ist. Ein zur Ausführung angemeldetes Programm (Task) wird vom Scheduler gemäß seiner Priorität an die entsprechende Ebene der Auftragsliste angekettet. An einer Ebene können mehrere Tasks mit gleicher Priorität angemeldet sein. Man bezeichnet eine solche Softwareprioritätsebene auch als Pseudoprozessor. Der Dispatcher arbeitet auf die Auftragsliste und vergibt die CPU an die rechenbereite Task in der Ebene mit der höchsten Priorität. Eine Task kann folgende Zustände einnehmen:

- aktiv: Der Rechnerkern bearbeitet gerade eine Anweisungsfolge der Task
- bereit: Die Task ist arbeitsfähig, besitzt aber nicht den Rechnerkern
- wartend: Eine Task wartet auf ein Ereignis

Die Anzahl und Art der Tasks in der Auftragsliste und ihre Zustände charakterisieren zu einem wesentlichen Teil das Systemverhalten.

2. Verfahren zur Messung von Betriebssystemen

2.1 Allgemeines

Der Begriff Messung kann definiert werden als das Sammeln von quantitativen Daten über das Verhalten eines Systems. Wegen ihrer Komplexität sind die Betriebssysteme von Rechenanlagen besonders schwierig zu messen. Aber gerade bei Betriebssystemen sind quantitative Angaben über das Systemverhalten von großer Bedeutung, weil es in den meisten Fällen unmöglich ist durch qualitative Betrachtungen Rückschlüsse auf das richtige Arbeiten des Betriebssystems zu ziehen. Oft arbeitet ein Betriebssystem ein Benutzerprogramm korrekt ab, ist aber sehr ineffizient in der Ausnutzung der Betriebsmittel des Systems. Cantrell /1/ beschreibt ein Meßsystem, welches in dem gemessenen Betriebssystem Engpässe fand, die zwei Jahre lang einen Leistungsverlust von 30 % bewirkt hatten, ohne daß es jemandem aufgefallen war.

Für die Ermittlung von quantitativen Aussagen über ein Betriebssystem sind zwei Vorgehensweisen möglich. Einmal kann das System in einem Modell nachgebildet werden und dann das Verhalten dieses Modells untersucht werden, um daraus Rückschlüsse auf das Verhalten des realen Systems zu ziehen. Das Modell kann in einfachen Fällen als analytisches Modell oder bei komplexeren Systemen als Simulationsmodell realisiert werden. Modelle werden vornehmlich zur Leistungsvorhersage bei der Entwicklung von Betriebssystemen verwandt. Sie sollen im Rahmen dieses Berichts nicht näher behandelt werden. Die zweite Möglichkeit des Vorgehens besteht darin, am im Betrieb befindlichen System Messungen vorzunehmen und aus den Meßwerten Aussagen über das Verhalten der Leistungskenngrößen des Systems abzuleiten. Die Meßmethoden für Computersysteme können in zwei Klassen eingeteilt werden:

(1) Hardware-Meßverfahren und
(2) Software-Meßverfahren.

2.2 Hardware-Meßverfahren

Hardware-Meßverfahren sind Meßverfahren, die mit Hilfe einer elektronischen Meßapparatur über in den Werken des Objektrechners vorhandene Meßpunkte die Aktivitäten und die Zustände des Rechners in Form von Signalen und Potentialen aufnehmen. Die Reduktion der Meßdaten zu Meßergebnissen kann durch Software erfolgen.

Hardware-Meßverfahren beeinflussen den Aktivitätsverlauf im Objektrechner nicht. Sie sind sehr gut geeignet für Probleme, die Messungen auf einem niedrigen Ablaufniveau des Objektrechners bedingen, z.B. die Messung der Ausführung von Befehlen auf der Befehlsebene des Objektrechners. Sie können jedoch keine Aktivitäten identifizieren, die aus diskreten Programmzuständen herrühren, da diese aus den Zuständen der Hardware des Rechnersystems nicht als verschieden erkannt werden können. Hardware-Meßsysteme erfordern Anschlußpunkte am Objektrechner. Sie sind daher nicht ohne weiteres auf jedem Rechner einsetzbar.
Die Set-Up-Zeit für das Meßsystem ist verhältnismäßig hoch, ebenso der Aufwand für die Wartung. Jedesmal, wenn andere Funktionen gemessen werden sollen, müssen andere Anschlußpunkte ausgewählt werden bzw. muß die Schnittstellenverdrahtung geändert werden. Deswegen sind Hardware-Meßverfahren relativ unflexibel.

2.3 Software-Meßverfahren

Software-Meßverfahren sind Meßverfahren, die über programmierte Meßpunkte Ereignisse und Zustandsgrößen des Betriebssystems mit Programmen aufnehmen.

Für Software-Meßverfahren sind keine speziellen technischen Voraussetzungen nötig. Sie lassen sich prinzipiell auf jeder Rechenanlage durchführen. Mit ihnen ist die Messung von beliebigen Größen des Objektrechnersystems möglich, wie z.B. die Häufigkeit der Benutzung von Programmmoduln, die Länge von Warteschlangen etc.. Die Messungen können leicht modifiziert werden, da Meßpunkte und Programmteile für die Messungen programmiert ausgetauscht werden können. Software-Meßverfahren können ohne großen Aufwand für die Installation eingesetzt werden.
Es gibt allerdings auch Einschränkungen für den Einsatz von Software-Meßverfahren:

- Da das Meßsystem sich im Speicher des Objektrechners befindet, muß es mit allen anderen Programmen, die auf dem Rechner laufen, insbesondere mit dem Betriebssystem selber, koexistieren. Daraus ergeben sich einschränkende Randbedingungen beim Entwurf des Meßsystems
- Die Messungen sind mit einer Störung des normalen Aktivitätsverlaufes im Objektrechner verbunden. Diese Störung fällt aber erst dann ins Gewicht, wenn die Dauer der Messungen nicht mehr vernachlässigbar klein gegenüber der Ausführungsdauer der gemessenen Objektprogramme ist.
- Da Software-Meßsysteme an die Umgebung des Objektrechnersystems angepaßt werden müssen, sind sie normalerweise nicht ohne Änderungen auf andere Rechnersysteme übertragbar.

Es gibt mehrere Techniken der Realisierung von Software-Meßverfahren:

(1) Zugeordneter Monitor-Computer
Ein anderer Rechner wird benutzt, um einen Botschaften-Verkehr zum Objektrechnersystem zu simulieren. Da der sendende Rechner auch die Antworten auf die Botschaften empfängt, kann er Statistiken über den Durchsatz und die Antwortzeit des Objektrechnersystems aufstellen. Da der zugeordnete Rechner nicht direkt in das Objekt-Rechnersystem eingreifen kann, können interne Variable des Objektrechnersystems nicht gemessen werden. Diese Technik ist daher nur begrenzt brauchbar.

(2) Übergeordnetes Monitor-System
Es behandelt das zu messende Betriebssystem als ein Problemprogramm, welches unter seiner Kontrolle abläuft. Mit dieser Technik können sehr detaillierte Informationen über das Verhalten des gemessenen Betriebssystems gewonnen werden. Sie verursacht allerdings auch ein hohes Maß an Leistungsverlust des gemessenen Betriebssystems.

(3) Parallel laufendes Meßsystem
Das Meßsystem läuft zusammen mit dem zu messenden Betriebssystem auf dem Objektrechner. Es ist durch ein für das Betriebssystem logisch unsichtbares Interface angeschlossen und sammelt über dieses Interface die Meßdaten auf. Diese Lösung ist effektiv und liefert alle relevanten Meßdaten.

Das in den folgenden Abschnitten beschriebene Meßsystem arbeitet nach dem letzteren Verfahren.

3. Das Softwaremeßsystem MECROS

Wie gezeigt wurde sind Softwaremeßverfahren flexibler als Hardwaremeßverfahren und leichter zu installieren. Aus diesem Grunde wurde für die Messungen an CALAS7O ein Softwaremeßsystem entwickelt. Das Softwaremeßsystem läuft parallel zu CALAS7O auf dem Objektrechner.

3.1 Leistungsanforderungen und Systemkonfiguration

An dieses System wurden die folgenden Leistungsanforderungen gestellt:

(1) Leichte und lexible Handhabung des Meßsystems
Der Start des Meßsystems und die Anwahl von Messungen müssen bedienungsfreundlich sein.

(2) Erweiterbarkeit der Messungen
Neue Meßfunktionen müssen leicht in das System eingebracht werden können.

(3) Online-Darstellung von Meßdaten
Ausgewählte Meßdaten sollen während eines Meßlaufs auf einem Display beobachtet werden können.

(4) Bequeme Auswertung
Der Benutzer muß eine einfache und klar definierte Schnittstelle für spezielle Auswerteprogramme vorfinden. Die Auswertung muß offline auf einem Großrechner vorgenommen werden können.

(5) Minimale Leistungsminderung des Objektrechnersystems
Die Messungen sollen den Objektrechner nur unwesentlich belasten, um die Leistungsfähigkeit des Betriebssystems zu erhalten und um die Messung selber nicht zu verfälschen.

(6) Störsicherheit des Objektrechnersystems
Während der Installation und der Benutzung des Meßsystems darf das auszumessende Betriebssystem weder angehalten noch in seiner logischen Funktion geändert werden.

Die Konfiguration des Gesamtsystems zeigt Bild 1. Wesentliche Merkmale sind:

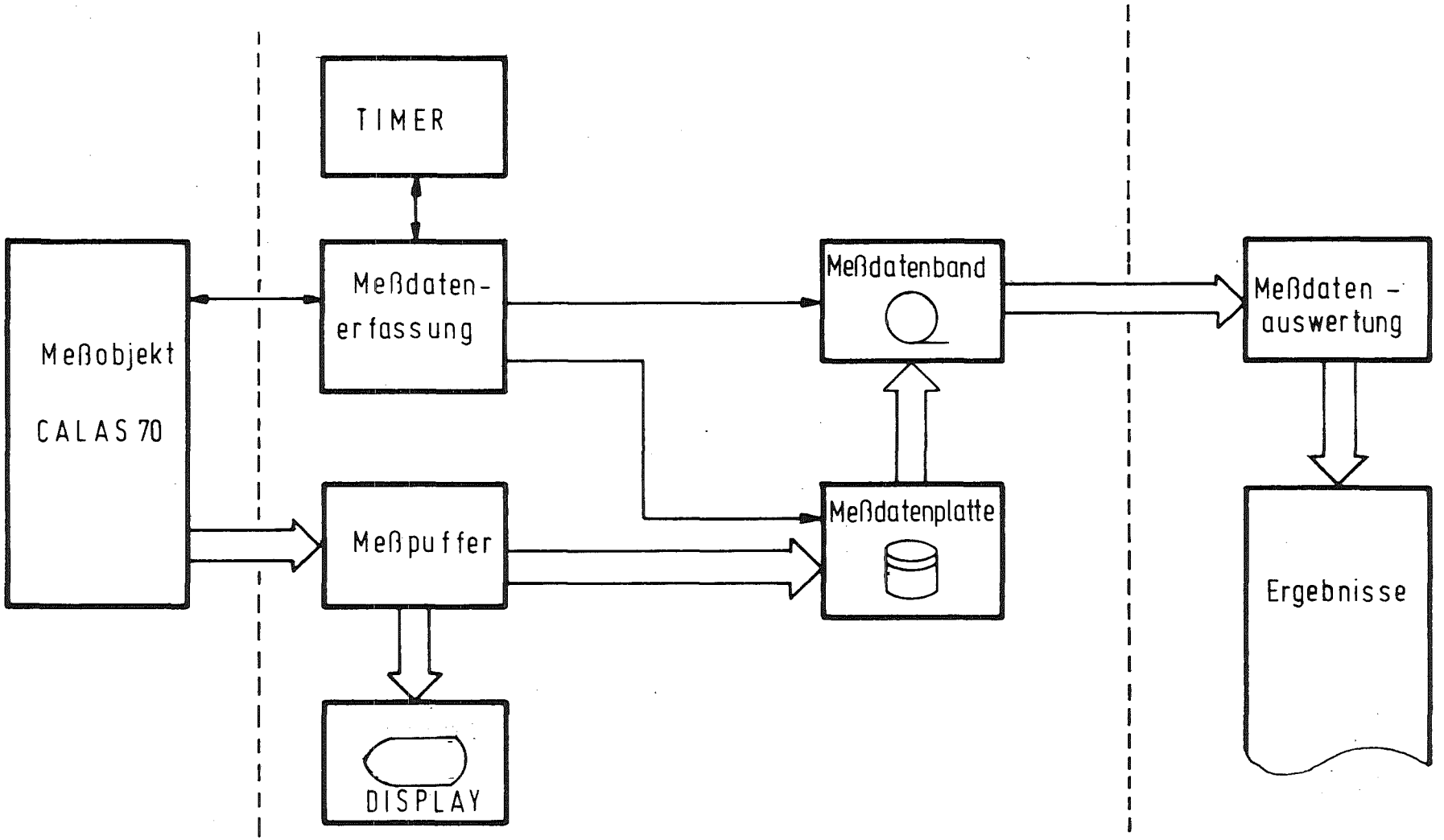
(1) Das Meßsystem läuft auf dem Objektrechner TR86 zusammen mit dem zu messenden Betriebssystem CALAS7O. Die Meßdatenauswertung, auf die im Rahmen dieses Berichtes nicht näher eingegangen werden soll, läuft offline auf einer IBM37O.
Die Auslagerung der Meßdatenauswertung auf den Großrechner hat zwei Vorteile:

- Der Objektrechner wird nicht mit der Auswertung der Meßdaten belastet
- Für die Entwicklung von Meßdatenauswerteprogrammen steht der Komfort eines Großrechners zur Verfügung.

(2) Abspeicherung der Meßdaten in 3 Speicherebenen
Der Meßdatenspeicher umfaßt 3 Speicherebenen:
- den Meßpuffer im Kernspeicher
- die Meßdatenplatte
- das Meßdatenband, über welches die Daten an die Auswertungsprogramme übergeben werden

Durch diesen hierarchischen Aufbau der Meßdatenspeicher werden Geschwindigkeit und Kapazität der einzelnen Speichermedien günstig für die Meßdatenabspeicherung ausgenutzt.

(3) Das Meßdaten-Display
Auf dem Meßdaten-Display können die Werte von Meßgrößen während des Meßlaufs dargestellt werden. Dadurch hat der Benutzer die Möglichkeit Meßdaten während des Meßlaufs direkt zu sehen, ohne die Ergebnisse der Auswerteprogramme abwarten zu müssen. Der Informationslauf im Meßsystem umfaßt somit zwei Schleifen /Bild 2/. Der Informationsfluß über das Display ist schnell und liefert einen groben Überblick über das Verhalten des gemessenen Betriebssystems. Der Informationsfluß über das Meßdatenband und die Auswerteprogramme ist langsam, liefert aber die ge-



Datenfluß

Kontrollfluß

Bild 1 MECROS Systemkonfiguration

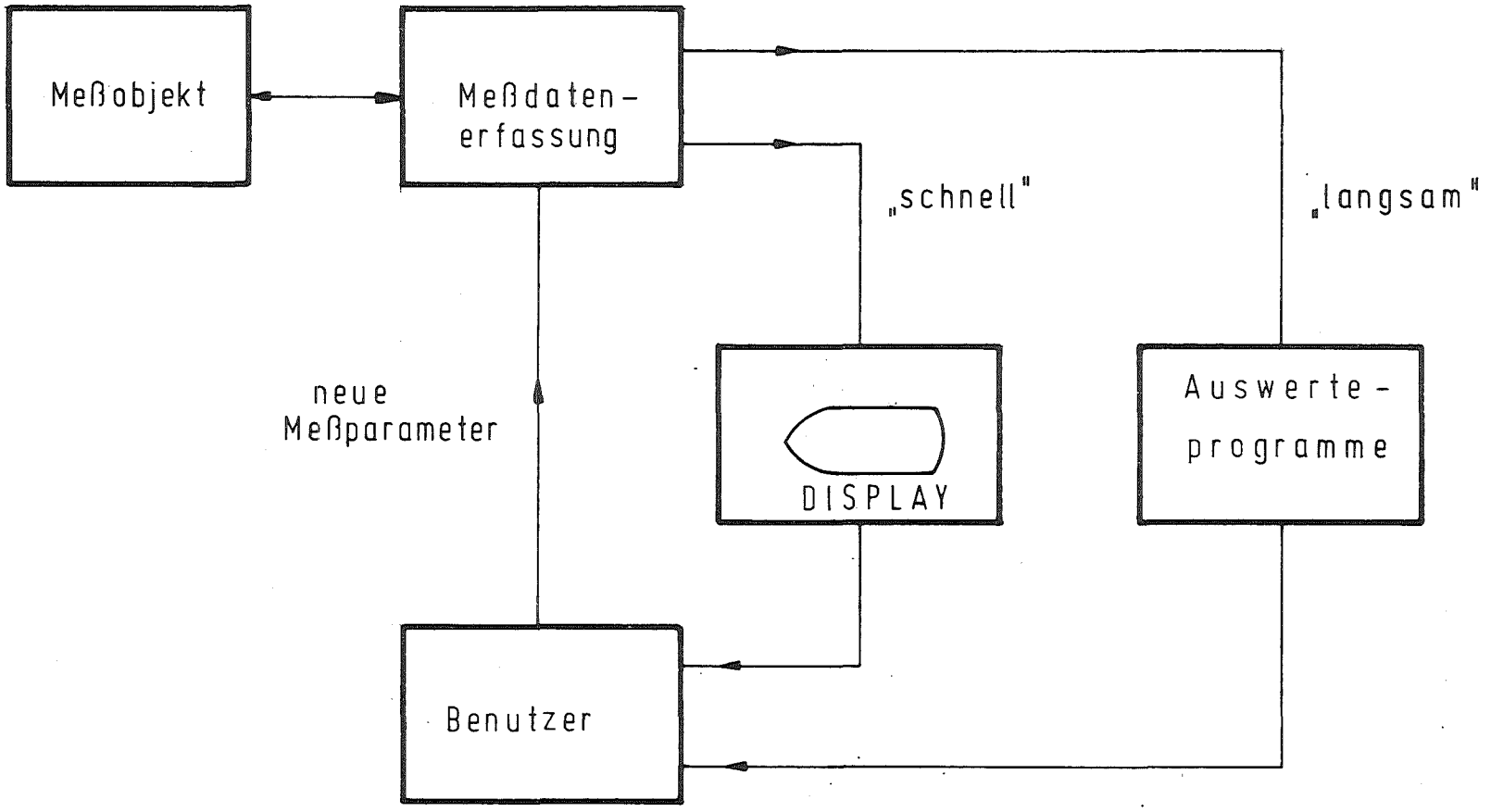


Bild 2 Informationsfluß im Meßsystem

nauen Ergebnisse der Messung, die dann einer beliebigen Verarbeitung unterzogen werden können.

Die Erreichung eines Meßzieles kann so als iterativer Prozeß aufgebaut werden. Auf der ersten Stufe dieses Prozesses verschafft sich der Benutzer mit Hilfe des Informationsflusses über das Display einen Überblick über das Verhalten des Betriebssystems. Er ändert dann gegebenenfalls solange die Meßfunktionen,bis er sicher ist,durch die Messungen die signifikaten Werte zu erhalten. Dann wird auf der nächsten Stufe der Informationsfluß über das Meßdatenband und die Auswerteprogramme dazu benutzt, um exakte und archivierungsfähige Aussagen zu erhalten.

3.2 Organisation und Steuerung des Meßsystems

Die Aktivierung des Meßsystems und die Steuerung der Messungen wird im Wesentlichen von den Moduln Initailisierung, Meßsteuerung und Meßdatenspeichersteuerung durchgeführt /Bild 3/.
Man kann das Meßsystem als virtuelle Maschine ansehen, welche drei Zustände hat:

(1) schlafend
(2) bereit
(3) aktiv

Die Zustände und die möglichen Übergänge sind in Bild 4 dargestellt.

Im Zustand "schlafend" liegt das Meßsystem als Lademodul in einer Benutzerdatei. Es enthält alle auswählbaren Meßfunktionen. Jede Meßfunktion ist durch eine Meßroutine repräsentiert. Die im Meßsystem enthaltenen symbolischen Referenzen zum Meßobjekt, also zum Betriebssystem CALAS7O, wurden vorher durch einen Bindelauf mit den Objektmoduln von CALAS7O aufgelöst. Es ist sinnvoll alle Referenzen zum Meßobjekt symbolisch zu adressieren, gegebenenfalls durch zusätzliche Einführung von Namen im Meßobjekt, weil das Meßsystem damit unabhängig von Verschiebungen wird, die sich durch Änderungen im Meßobjekt ergeben können.

Der Zustand "bereit" wird durch Aufruf eines Initialisierungsprogramms unter Angabe des den Meßlauf spezifizierenden Meßparametersatzes erreicht. Das Meßsystem wird zusammen mit dem Meßparametersatz in einen reservierten Adreßraum geladen. Mit Hilfe von Interface-Tabellen wird die Schnittstelle zu den benötigten Betriebsmitteln des Meßobjekts generiert. Die Anschlüsse an die Meßpunkte werden gemäß den Spezifikationen des Meßparametersatzes hergestellt.
Alle Aktivitäten, die während des Überganges in den Zustand "bereit" ausgeführt werden, laufen im Hintergrund mit niedriger Priorität ab, so daß der Objektrechner bei der Ausführung von zeitkritischen Aufträgen nicht gestört wird.

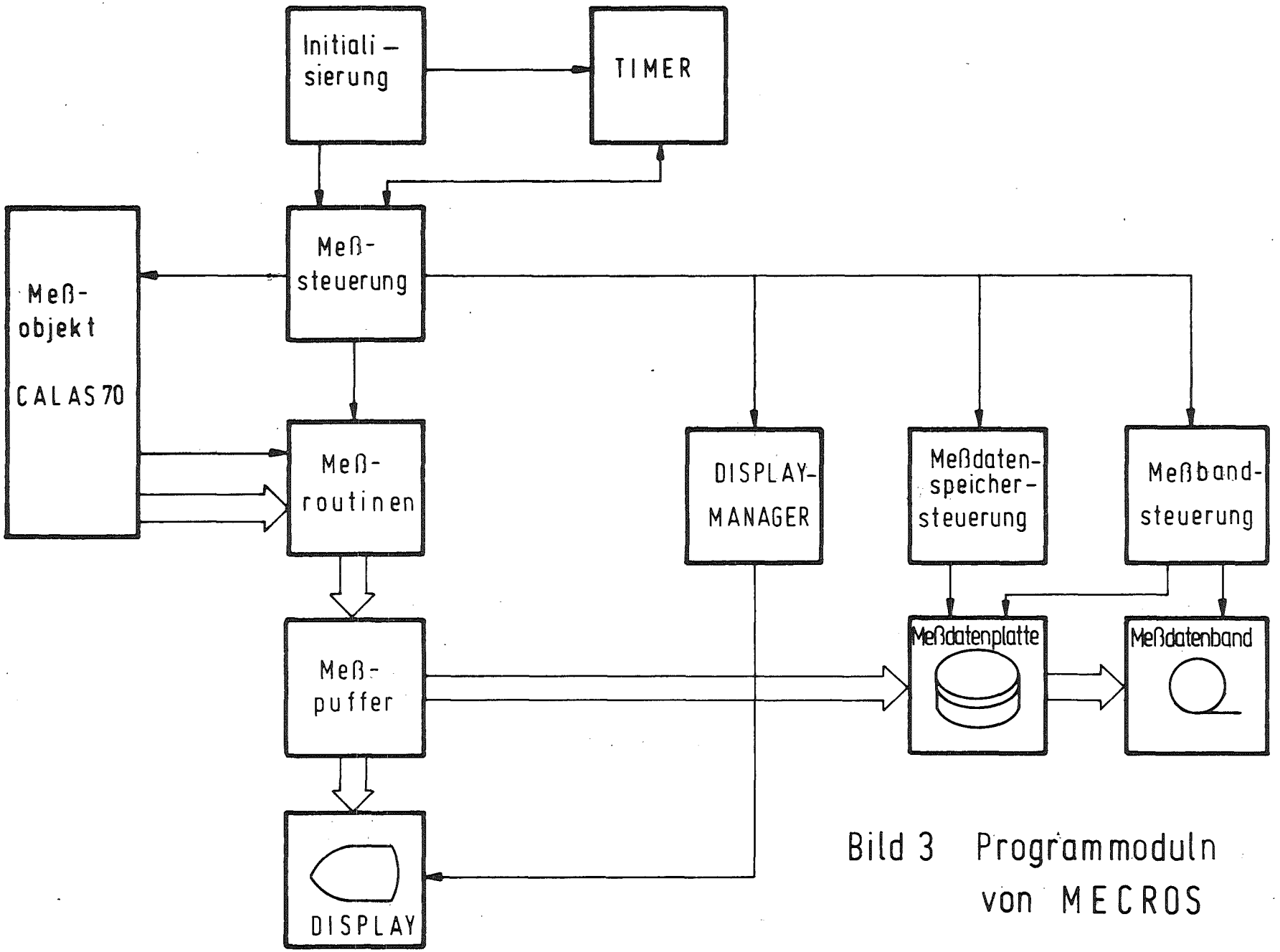


Bild 3 Programmoduln von MECROS

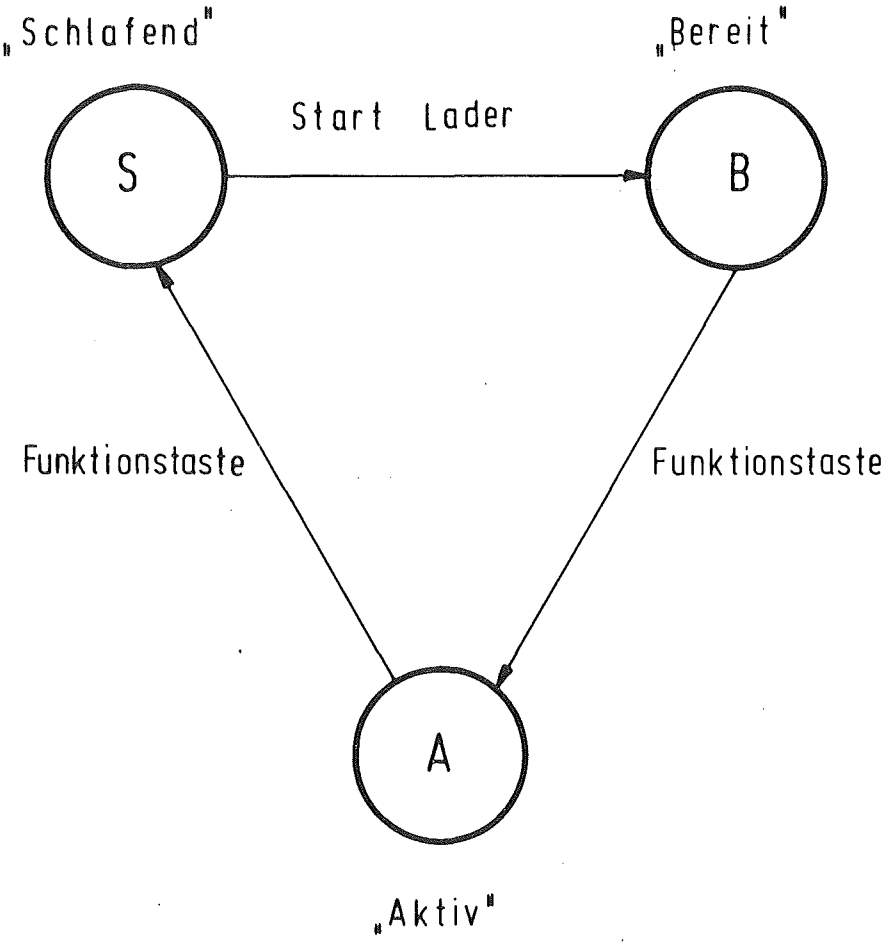


Bild 4 Systemzustände von MECROS

Der Zustand "aktiv" entspricht dem eigentlichen Meßlauf. Er wird durch Betätigen einer programmierten Funktionstaste auf der Konsole erreicht. Im Zustand "aktiv" werden die durch den Meßparametersatz angewählten Messungen durchgeführt, alle Meßdaten abgespeichert und ausgewählte Meßdaten auf dem Display gezeigt.

Der Zustand "aktiv" wird durch erneute Betätigung der Funktionstaste oder durch Überlauf des Meßdatenbereichs auf der Meßplatte verlassen. Beim Verlassen des aktiven Zustandes wird die Schnittstelle zwischen Meßobjekt und Meßsystem abgebaut. Das Meßobjekt ist damit wieder im ursprünglichen unveränderten Zustand. Der Meßlauf ist beendet und das Meßsystem bedindet sich wieder im Zustand "schlafend".

Der Benutzer hat jetzt wahlweise die Möglichkeit die Meßdaten auf der Meßplatte für eine spätere Auswertung auf dem Meßband zu archivieren oder sie zu verwerfen. Danach kann das Meßsystem erneut gestartet werden.

3.3 Die Schnittstelle für die Messungen

Es wurde bereits erwähnt, daß bei einem Softwaremeßsystem zwangsläufig eine Beeinflussung des Meßobjekts durch das Meßsystem auftritt. Diese Beeinflussung liegt darin begründet, daß das Meßsystem nicht vom Meßobjekt entkoppelt werden kann. Die Kopplung zwischen Softwaremeßsystem entsteht, weil das Softwaremeßsystem selbst über keinerlei Betriebsmittel verfügt und deswegen die Betriebsmittel des Objektrechnersystems mitbenutzen muß. Das Softwaremeßsystem benötigt Betriebsmittel für die Aufnahme der Meßdaten aus dem Meßobjekt, wie CPU und Kernspeicher, und Betriebsmittel für die Abspeicherung der Meßdaten, wie Kanäle und externe Speicher. Die Mitbenutzung der Betriebsmittel muß effektiv durchgeführt werden, damit die Belastung des Objektrechners durch das Meßsystem vernachlässigbar ist.
Für die Messung von internen Größen eines Betriebssystems mit einem Software-Meßverfahren gibt es grundsätzlich zwei Meßmethoden:

(1) ereignisgesteuerte Messungen
(2) zeitgesteuerte Messungen

Das hier vorgestellte System enthält beide Meßmethoden.

3.3.1 Ereignisgesteuerte Messungen

Ereignisgesteuerte Messungen werden durch ein Ereignis im Betriebssystem ausgelöst. Bei Eintritt des Ereignisses wird die Kontrolle vom Betriebssystem an die dem Ereignis zugeordnete E-Meßroutine abgegeben.
In der Meßroutine wird nun die eigentliche Messung durchgeführt: z.B. das Zählen der Häufigkeit des Ereignisses, die Aufnahme des aktuellen Wertes von relevanten Zustandsvariablen etc.. Anschließend wird die Kontrolle wieder an das Betriebssystem zurückgegeben. In der Meßroutine sind nur lesende Zugriffe auf die Daten des Betriebssystems erlaubt. Registerinhalte sind gegebenenfalls am Anfang der Meßroutine zu retten

und am Ende wieder einzusetzen. Damit ist gewährleistet, daß durch die Aktivitäten in der Meßroutine keine Daten im Betriebssystem verändert werden.

Für die Realisierung eines Meßpunktes (engl. "Probe" oder "Hook") im Betriebssystem bildet sich bei Maschinen mit einfacher Hardwarestruktur die Unterprogrammtechnik an. Bei Eintritt des Ereignisses wird aus der dem Ereignis zugeordneten Codefolge im Meßobjekt die entsprechende Meßroutine im Meßsystem als Unterprogramm aufgerufen.

Die Implementierung geschieht wie folgt: Im Betriebssystem wird an geeigneter Stelle in der dem Ereignis zugeordneten Codefolge der Originalcode gegen einen CALL SUBROUTINE in die entsprechende Meßroutine ausgetauscht. Die Meßroutine wird durch RETURN abgeschlossen. Der ausgetauschte Originalcode des Betriebssystems wird in die Meßroutine kopiert und dort ausgeführt. Der Codeaustausch geschieht dynamisch mit Hilfe einer PROBE-Tabelle. Hierbei wird der Meßpunkt zu Beginn des Meßlaufs generiert und am Ende des Meßlaufs durch Wiedereinfügen des Originalcodes wieder abgebaut /Bild 5/. Die dynamische Generierung der Meßpunkte gewährleistet einen störungsfreien Aufbau der Schnittstelle Meßobjekt-Meßsystem für die Messungen.

Es ist wichtig auf eine effiziente Codierung der Meßroutinen zu achten, um die Verfälschung des Meßobjekts, die durch das Einfügen der Meßroutinen entsteht, so klein wie möglich zu halten.

Ereignisgesteuerte Messungen sind sehr vielseitig verwendbar. Sie können exakte Aussagen über alle Zustände und Zustandsfolgen, die das Betriebssystem durchläuft liefern. Mit ihnen kann also das dynamische Verhalten eines Betriebssystems vollständig erfaßt werden.

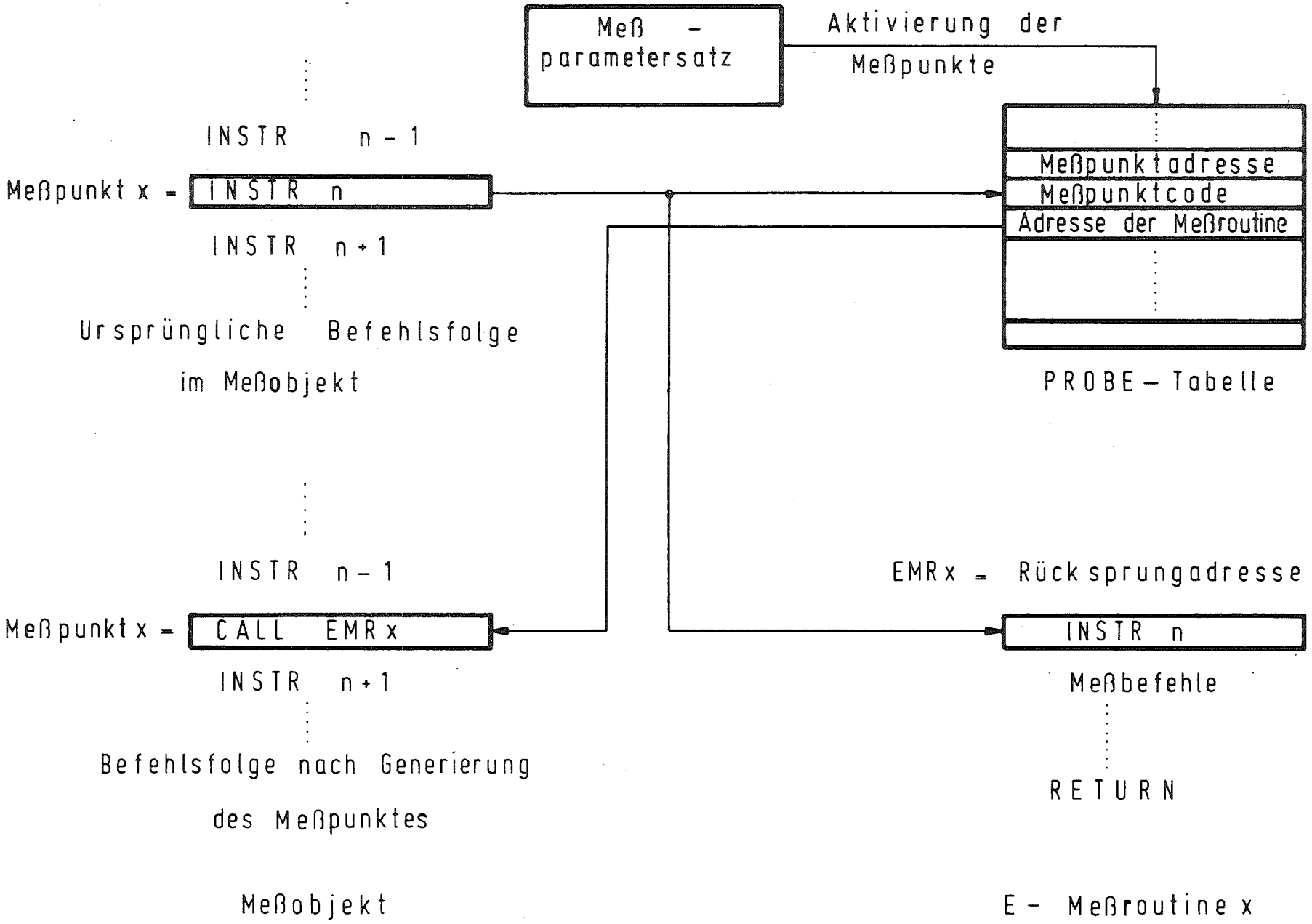


Bild 5 Generierung eines Meßpunktes

3.2.2 Zeitgesteuerte Messungen

Zeitgesteuerte Messungen werden durch einen Uhr-Interrupt ausgelöst. Die Frequenz der Messungen ist frei wählbar. Nach Rettung des aktuellen Betriebssystemzustandes wird die Kontrolle an die Z-Meßroutine abgegeben. Alle aktiven Z-Meßroutinen werden ausgeführt. Die Aktivierung wird entsprechend den Angaben des Benutzers durch eine SWITCH-Tabelle gesteuert /Bild 6/. In einer Meßroutine können durch Zugriffe auf Listen des Betriebssystems Zustandsvariable des Betriebssystems gelesen werden. Nach Ausführung aller aktiven Meßroutinen wird der alte Zustand des Betriebssystems wieder eingesetzt und die Kontrolle an das Betriebssystem zurückgegeben.

Zeitgesteuerte Messungen ergeben eine gute statistische Aussage über das Verhalten des Betriebssystems, wenn die Messungen über einen genügend langen Zeitraum erfolgen.
Die Forderung, daß die Meßfrequenz stochastisch unabhängig von Ereignissen im Betriebssystem sein muß, kann trotz der festen Frequenz als hinreichend erfüllt angesehen werden, da das Auftreten von zur Meßfrequenz synchronen Ereignisfolgen nur eine sehr geringe Wahrscheinlichkeit hat.

Damit der normale Ablauf im Betriebssystem nicht gestört wird, sollte die Messung nur eine Zeit in Anspruch nehmen, die um eine Größenordnung kleiner ist, als der mittlere Abstand zwischen zwei externen Ereignissen.

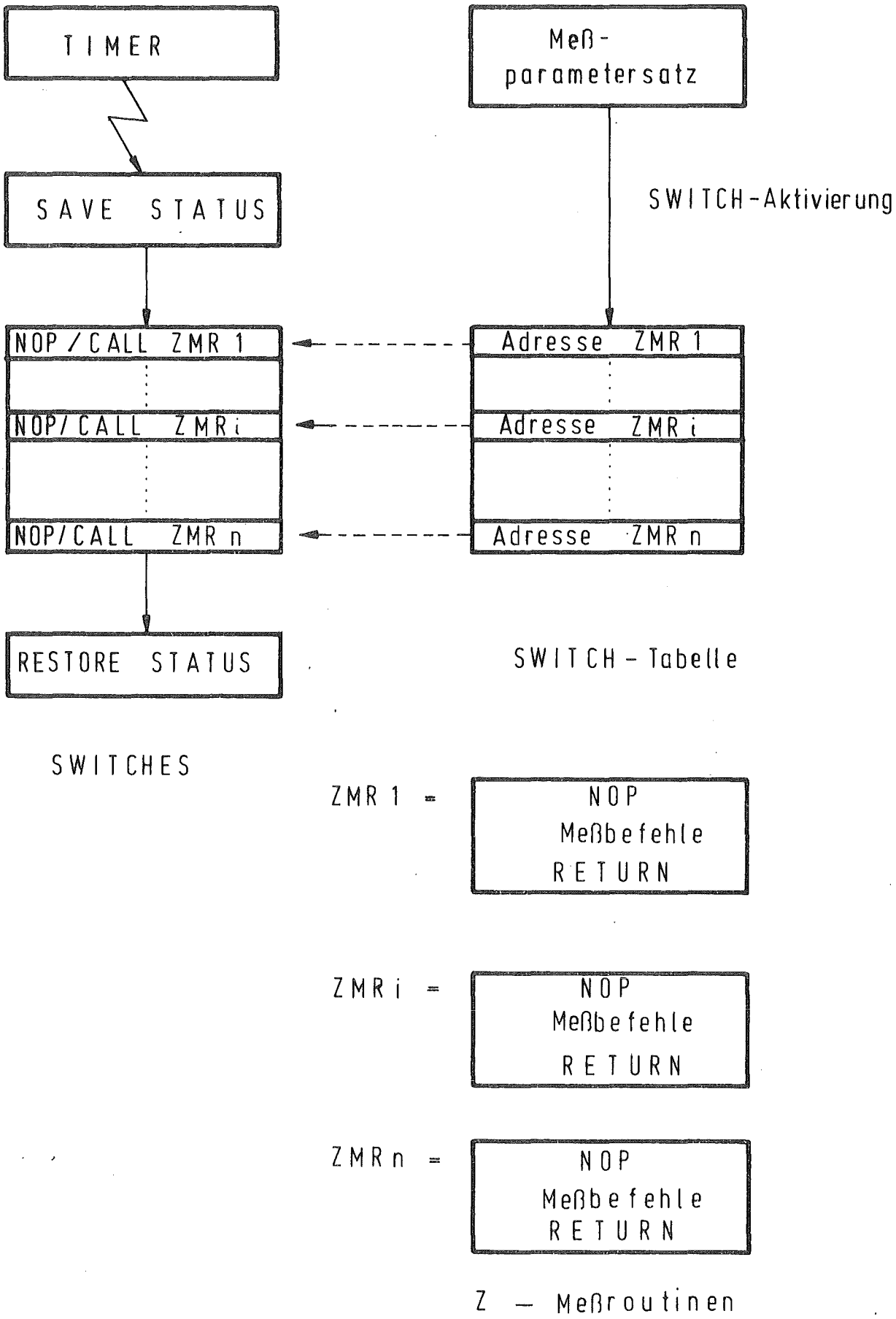


Bild 6 Aktivierung der zeitgesteuerten Messungen

3.4 Meßdatenabspeicherung

Die Meßdatenabspeicherung muß den nachstehenden Forderungen genügen:

(1) Die Meßdaten müssen schnell abgespeichert werden, damit der Objektrechner nur unwesentlich durch die Abspeicherung belastet wird.

(2) Die Kapazität des Meßdatenspeichers muß groß sein, um umfangreiche Messungen durchführen zu können.

(3) Die Meßdaten müssen so abgespeichert werden, daß sie von Auswertungsprogrammen leicht interpretiert werden können.

Die gewählte Organisation der Meßdatenabspeicherung hat folgende Struktur:

- Die Abspeicherung der Meßdaten erfolgt in drei Speicherebenen,

  o dem Meßpuffer im Arbeitsspeicher
  o dem Meßdatenfile auf der Meßdatenplatte
  o dem Meßdatenband

  Dadurch können die Anforderungen sowohl an die Geschwindigkeit der Abspeicherung als auch an die Kapazität des Meßdatenspeichers erfüllt werden.

- Sämtliche Informationen, die der Interpretation der Meßdaten durch Auswetungsprogramme dienen, werden getrennt von den eigentlichen Daten in einem Beschreibungssatz gehalten. Dieser Beschreibungssatz wird zusammen mit den Meßdaten den Auswertungsprogrammen übergeben. Dadurch kann die Auswertung der Daten leicht erfolgen. Das hat zudem den Vorteil, daß der Meßpuffer direkt als Display-File für das Meßdatendisplay benützt werden kann.

Im Einzelnen ist die Meßdatenabspeicherung durch die folgenden Komponenten realisiert:

- Meßdatensatz
  Jeder aktiven Meßroutine ist ein Meßdatensatz zugeordnet. Er hat eine variable Anzahl von Feldern. Ihre Anzahl richtet sich nach der Anzahl der Daten, die in einer Meßroutine aufgenommen werden.

- Beschreibungssatz
  Jeden Meßdatensatz ist ein Beschreibungssatz zugeordnet. Er enthält alle für die Interpretation der Meßdaten durch Auswertungsprogramme benötigten Informationen.

- Meßdatenblock
  Ein Meßdatenblock besteht aus den Meßdatensätzen aller aktiven Meßroutinen. Er wird jeweils am Ende eines Meßintervalls*) aus dem Meßdatenpuffer im Arbeitsspeicher in das Meßdatenfile auf der Meßplatte übertragen.

- Beschreibungsblock
  Er enthält die Beschreibungssätze für alle aktiven Meßdatensätze und wird am Beginn eines Meßlaufs an den Anfang des Meßdatenfiles geschrieben.

- Meßdatenfile
  Das Meßdatenfile enthält alle in einem Meßlauf aufgenommenen Meßdatenblöcke und den zugehörigen Beschreibungsblock. Es steht auf einer dem Meßsystem zugeordneten Platte.

- Meßdatenband
  Das Meßdatenband enthält einen oder mehrere Meßdatenfiles. Ein Meßdatenfile wird am Ende eines Meßlaufs auf das Meßdatenband geschrieben. Das Meßdatenband dient als Eingabeträger für Auswerteprogramme.

---

*) Das Meßintervall bestimmt also die Auflösung der Messungen. Es ist identisch mit dem Zeitintervall für die zeitgesteuerten Messungen

Um das zu messende Betriebssystem durch die Meßdatenabspeicherung so wenig wie möglich zu belasten ist dem Meßsystem in der vorliegenden Implementierung eine Platteneinheit mit eigenem sehr schnellen Geräte- und Kanaltreiber exklusiv zugeordnet. Das Interface des Meßsystems zu dieser Meßplatte wird dynamisch zu Beginn eines Meßlaufs generiert und am Ende eines Meßlaufs wieder abgebaut. Der Normalbetrieb wird also nicht beeinträchtigt.
Dem Benutzer steht die den Messungen zugeordnete Platte während eines Meßlaufs nicht zur Verfügung. Benutzerdateien auf dieser Platte sind vor Beginn der Messungen gegebenenfalls auf andere Platteneinheiten zu kopieren.
Die Übernahme eines Meßdatenfiles von der Meßplatte auf das Meßband läuft als normales Benutzerprogramm im Hintergrund mit geringer Priorität ab und belastet daher das Objektrechnersystem nicht.

3.5 Die Belastung des Meßobjekts durch das Meßsystem

Der Benutzer muß sich bei einem Einsatz des Meßsystems darüber im Klaren sein, inwieweit seine Messungen den Objektrechner belasten und wie daher die Messungen selber durch diese Belastung verfälscht werden. Die Belastung des Objektrechners durch das Meßsystem kann in drei Komponenten aufgeteilt werden /Bild 7/:

(1) Die erste Komponente wird durch die Meßdatenabspeicherung verursacht. Sie ist abhängig von der Meßintervallfrequenz, aber in jedem Fall vernachlässigbar klein.

(2) Die zeitgesteuerten Messungen bilden eine Belastung, die proportional zur Meßintervallfrequenz f_i und zur Anzahl n der pro Meßintervall in den zeitgesteuerten Messungen aufgenommenen Datenworte ist.
Ihr Wert ist ungefähr $6 \cdot 10^{-4} \cdot n \cdot f_i$ % .

(3) Die dritte Lastkomponente wird durch die ereignisgesteuerten Messungen hervorgerufen. Sie ist abhängig von der Anzahl der gemessenen Ereignisse pro Zeileneinheit und damit von der Anzahl der aktiven Meßpunkte und der Häufigkeit des Auftretens von zu messenden Systemereignissen. Die Systemereignisse werden im Wesentlichen durch die Aktivitäten des Systemkerns hervorgerufen. Damit ist die Belastung durch die ereignisgesteuerten Messungen direkt proportional zum Overhead des Betriebssystems. Durch Messungen ergab sich bei dem vorliegenden System im Mittel eine Belastung pro Meßpunkt von 2% des Betriebssystemoverheads.

Soll die Belastung des Objektrechners durch die Messungen einen vorgegebenen Wert nicht überschreiten, so ist es erforderlich die Anzahl der zeitgesteuerten Messungen mit der Meßintervallfrequenz Z geeignet abzustimmen und die Anzahl der ereignisgesteuerten Messungen an das jeweilige Aufgabenprofil anzupassen.

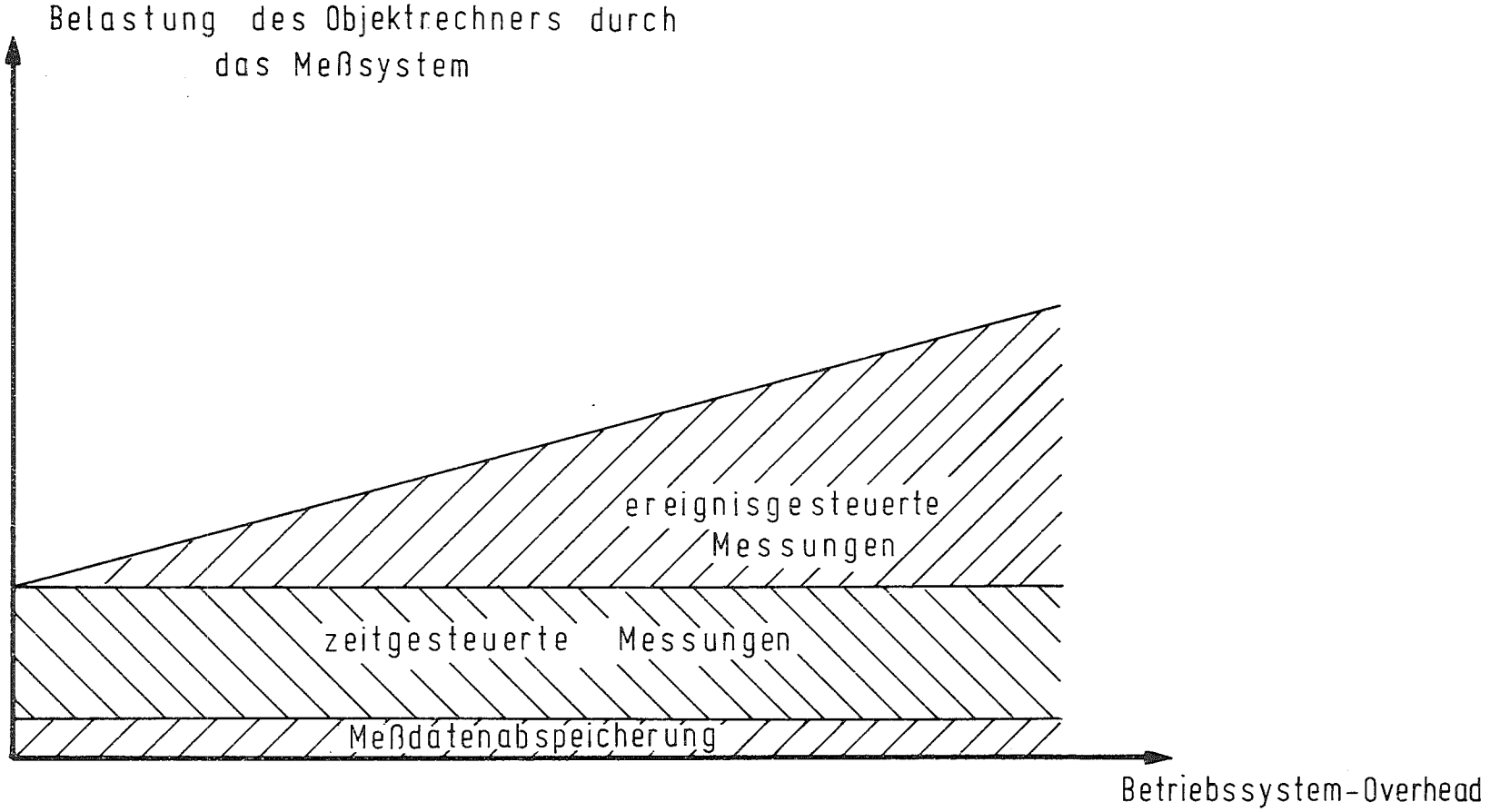


Bild 7 Komponenten der Belastung des Objektrechners durch das Meßsystem

4. Aufbau und Aufruf von Messungen in MECROS

In diesem Abschnitt sollen der Aufruf von Messungen und die notwendigen Angaben zum Aufbau von Messungen mit MECROS gezeigt werden.

Aufruf von Messungen

Der Aufruf von MECROS für einen Meßlauf und die Generierung eines Meßparametersatzes zur Auswahl der gewünschten Messungen erfolgen nach den Konventionen von CALAS7O für den Aufruf von Benutzerprogrammen und die Generierung von Parametersätzen*).

Aufruf:

```
STA, MECROS, PAR1;
```

Der Aufruf enthält den Programmnamen und den Namen des gewünschten Meßparametersatzes.

Meßparametersatz:

Der Meßparametersatz wird interaktiv am Display erstellt. Er enthält die Namen der gewünschten Messungen (max. 8 Character). Das Prefix 'DISP' vor einem Namen gibt an, daß die jeweilige Messung auf dem Display darzustellen ist.

Beispiel eines Meßparametersatzes:

```
DTX, PAR1;
*     Ø5Ø
ØØØØ "DISPQUEU"
ØØ1Ø "FREQUE"
ØØ2Ø "SOURCE"
ØØ3Ø "PRIO";
```

---

*) Wurden vom Benutzer neue Meßroutinen mit neuen Referenzen auf das Meßobjekt eingefügt, so muß natürlich zuvor ein Bindelauf von MECROS mit den Moduln von CALAS7O durchgeführt worden sein.

Meßparameterliste:

Jede Messung ist durch einen Eintrag in die Meßparameterliste gekennzeichnet. Der Eintrag enthält u.a. die Kennzeichnung der Messung laut Meßparametersatz, einen Verweis auf den zugehörigen Bildtext in der Bildtextliste für die Displaydarstellung, die Anzahl der Meßdatenworte für die Messung im Meßpuffer, einen Zeiger auf die Verweistabelle, die Länge der Verweistabelle, sowie einen Zeiger auf den zugehörigen Eintrag in der Codeliste.

Beispiel:

```
 *
 *
 PARAMETERLISTE
PARLI.=''SOUR''              INTERRUPTQUELLEN          20773
       ''CE''
       AA INPARA.            INTERPRETATIONSPROGRAMM
       '000000'H             BILDTEXTVERWEISNR, ANF. DER MESSTABELLE
       7                     LAE. DER MESSTABELLE
       AA TSOURCE.           ANF. DER VERWEISTABELLE
       18                    LAENGE DER VERWEISTABELLE
       AA LICO1.             ANF. DER CODELISTE
       ''FREQ''              INTERRUPTFREQUENZ
       ''UE''
       AA INFREQ.
       '100000'H
       100
       AA TFREQU.
       1
       AA LICO2.
       ''QUEU''              WARTESCHLANGEN
       ''E''
       AA INPARA.
       '200000'H
       32
       AA TQUEUE.
       100
       AA LICO3.
```

Bei der Abbildung von Meßdaten in den Meßpuffer gibt es zwei Möglichkeiten:

(a) Die Abbildung erfolgt über Verweistabellen. Die Zuordnung der einzelnen Meßdatenwerte zu dem für diese Messung reservierten Meßpufferbereich geschieht über geeignete Angaben in der zugehörigen Verweistabelle. Als Interpretationsprogramm ist dann in der Meßparameterliste INPARA anzugeben.

(b) Die Abbildung der Meßdatenwerte in den Meßpuffer wird durch Indizes gesteuert, die aus der Messung selber bestimmt werden. In diesem Fall degeneriert die Verweistabelle zu einem leeren Wort. Als Interpretationsprogramm muß in der Meßparameterliste INFREQ angegeben werden.

Verweistabellen:

Verweistabellen dienen zur Abbildung von Meßdaten in den Meßpuffer und gegebenenfalls zur Abbildung von gestreuten Listen im Meßobjekt auf dichte Listen im Meßpuffer. Sie beschreiben durch positive ganze Zahlen die relative Anordnung der einzelnen Meßdatenworte im Meßpuffer. Bei der Abbildung von gestreuten Listen sind nicht belegte Listenplätze durch -1 gekennzeichnet.

Bei Steuerung der Abbildung durch aus der Messung selber gewonnenen Indizes degeneriert die Verweistabelle zu einem Wort (Inhalt Ø).

Beispiel:

```
 *
 *
 VERWEISTABELLE IT-QUELLEN
TSOURC.=-1                  PITE1:UNTERSPANNUNG
        -1                  PITE2:ALARM RECHNERKERNKANAL
        -1                  NICHT BELEGT
        0                   PITE4:STANDARDKANAELE
        1                   PITE5:KONSOLFERNSCHREIBER
        2                   PITE6:LOCHSTREIFENLESER
        3                   PITE7:LOCHSTREIFENSTANZER
        -1                  NICHT BELEGT
        -1                  NICHT BELEGT
        -1                  NICHT BELEGT
        -1                  NICHT BELEGT
        4                   PITE12:DATENKANAL
        -1                  NICHT BELEGT
        5                   PITE14:SICHTGERAETE
        6                   PITE15:BEFEHLSGEBER/TASTATUREN
        -1                  PITE16:KONSOLTASTE
        -1                  PITE17:ALARM UHR
        -1                  PITE18:UHR
 *
 *
 VERWEISWORT IT-FREQUENZ
TFREQU.=0
 *
```

Ereignisgesteuerte Messungen:

Codeliste:

Der Eintrag in der Codeliste enthält die Adresse des Meßpunktes im Meßobjekt, die Adresse der aufzurufenden E-Meßroutine und die Adresse für die Zwischenspeicherung des Originalcodes. Einem Meßparameter bzw. einer Meßroutine können mehrere Meßpunkte zugeordnet werden.

Beispiel:

```
 *
 *
 CODETABELLEN FUER MESSPARAMETER
LICODE.=!800000'H          ZEIGER AUF 1. AKTIVEN CODE    21000 21956
                            21968 21975 21997
 CODETABELLE FUER IT-QUELLEN-MESSUNG
LIC01.=AA INTMOD(501)      ADR. OBJEKTSYSTEM       20876
         SU SOURCE.        HOOKCODE
         AA HOOK1.         ADR. MESSSYSTEM
         '800000'H
 CODETABELLE FUER IT-FREQUENZ-MESSUNG
LIC02.=AA CAKERN(502)      20884
         SU FREQUE.
         AA HOOK2.
         '800000'H
 CODETABELLE FUER WARTESCHLANGENMESSUNG
LIC03.=AA CAKERN(503)      20892
         SU QUEUE1.
         AA HOOK3.
         AA CAKERN(504)
         SU QUEUE2.
         AA HOOK4.
         '800000'H
```

E-Meßroutinen:

Das folgende Beispiel zeigt den Aufbau einer E-Meßroutine. Jede E-Meßroutine ist als Unterprogramm organisiert. In der Meßroutine wird der Platz für die Zwischenspeicherung des Originalcodes aus dem Meßobjekt reserviert.

Beispiel:

```
 *
 *

 ---HOOKPROGRAMM ITQUELLEN---

(11)=SOURCE.=  0          21024 21714
HOOK1.=AA 0 U V           OBJEKTSYSTEMCODE SH 18        21025
        CA ZWI
        AA TSOURC. - 1    VERWEISTABELLE PIT-EBENEN
        AT '20000'H
        CA 2L
        CA 3L
        BE 0 U V          21708
        AA  1U
        CE  0 U V         21709
        B  ZWI
        SE SOURCE.
ZWI=0                     21705 21713
 *
 *
```

Zeitgesteuerte Messungen:

Codeliste:

Der Eintrag in der Codeliste für eine zeitgesteuerte Meßroutine enthält die Adresse i*) des Schalters (ZEITSW+i), wobei jede Meßroutine einen Schalter i belegt, den (Unterprogramm) Aufruf der Z-Meßroutine und die (DUMMY)-Adresse MESPEI.

Beispiel:

```
LIC05.=AA ZEITSW.+1        SCHALTER
          SU PRIREG.       E-MESSROUTINE
          AA MESPEI.       DUMMY-SPEICHER
```

Z-Meßroutine:

Jede Z-Meßroutine ist als Unterprogramm organisiert.

Beispiel:

```
PRIREG.=0
          B CAKERN(551)    SNEWPR
          CAE TPRIO.
          B CAKERN(17)     SOLDPR
          CAE TPRIO.+1
          B CAKERN(31)     SGITPR
          CAE TPRIO.+2
          SE PRIREG.
```

*) i=1...10 in der gegenwärtigen Implementierung

Bildtextliste:

Die Bildtestliste enthält die erklärenden Texte zu den Displaydarstellungen der Messungen. Die Einträge stehen in der Reihenfolge wie sie in der Bildtextverweisnummer der Meßparameterliste verlangt werden

Beispiel:

```
 *
 TEXTLISTEN
 *
TEXT1.=16                    22048
        '0A0A0A'H
        '0A0A0A'H
        '20200A'H
        'ANZ'I
        'AHL'I
        ' DE'I
        'R I'I
        'NTE'I
        'RRU'I
        'PTS'I
        '/IN'I
        'TER'I
        'RUP'I
        'TQU'I
        'ELL'I
        'E  'I
TEXT2.=15                    22049
        '0A0A0A'H
        '0A0A0A'H
        '20200A'H
        'VER'I
        'TEI'I
        'LUN'I
        'G D'I
        'ER 'I
        'INT'I
        'ERR'I
        'UPT'I
        'FRE'I
        'QUE'I
        'NZE'I
        'N  'I
```

5. Abschließende Bemerkungen

Das Meßsystem MECROS wurde mit simulierten Auftragsprofilen am Betriebssystem CALAS7O erprobt. Dadurch waren die Messungen reproduzierbar, was gerade bei den Tests des Systems sinnvoll war. Als besonders nützlich erwies sich die Möglichkeit, ausgewählte Meßdaten während des Meßlaufs auf dem Display darstellen zu können. Abgesehen von dem unmittelbaren Einblicken in komplexe Ablauffolgen im Betriebssystem war die Displaydarstellung vor allem auch eine große Hilfe beim Austesten des Meßsystems. Auf eine ausgiebige Kommentierung der Meßwerte auf dem Bildschirm wurde bei diesem experimentellen System verzichtet. Zusätzliche Textinformationen sind aber prinzipiell möglich und erhöhen natürlich die Lesbarkeit der Darstellungen wesentlich. Trotz der Einfachheit der Darstellungen konnten jedoch damit sehr wesentliche Informationen über das Betriebssystemverhalten gewonnen werden. Bild 8 und 9 zeigen dazu zwei Beispiele. In Bild 8 wird die aktuelle Länge der Warteschlangen pro Prioritätsebene, also die Anzahl der Tasks, die auf ihre Abarbeitung warten, dargestellt. Bild 9 zeigt die Anzahl von Aufträgen/Meßintervall, die von den in den einzelnen Prioritätsebenen aktiven Tasks abgesetzt werden. Beide Darstellungen charakterisieren bestimmte Zustände des Betriebssystems. Die Online-Verfolgung dieser Zustände am Display gibt einen schnellen und guten Überblick über das Betriebssystemverhalten unter verschiedenen Auftragsprofilen.

Im Meßsystem ist bereits eine umfangreiche Anzahl von Meßroutinen enthalten. Diese ist allerdings sicherlich nicht ausreichend, um alle für den jeweiligen Benutzer und die jeweilige Anwendung interessanten Aussagen über das Betriebssystem zu liefern. Der Benutzer kann sich jederzeit eigene Meßroutinen nach dem im Abschnitt 4 beschriebenen Schema aufbauen und in das Meßsystem integrieren. Der Aufbau von Meßroutinen ist hinreichend einfach. Durch die Erweiterung des Meßsystems um den Aufbau von Meßroutinen im Dialog wäre hier noch eine Verbesserung des Systems in Bezug auf größeren Be-

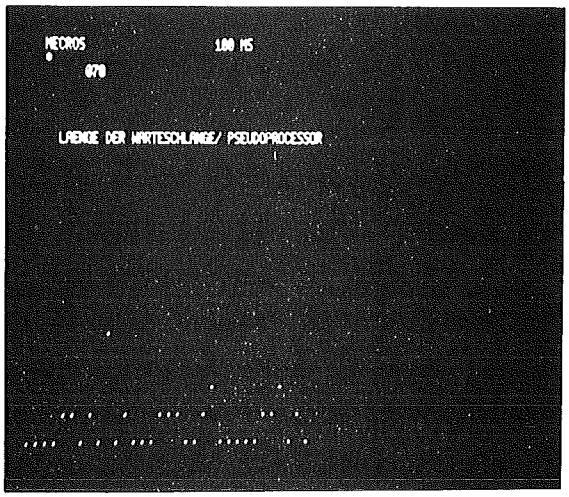


Bild 8 Auftragsliste: Anzahl der Tasks/Prioritätsebene

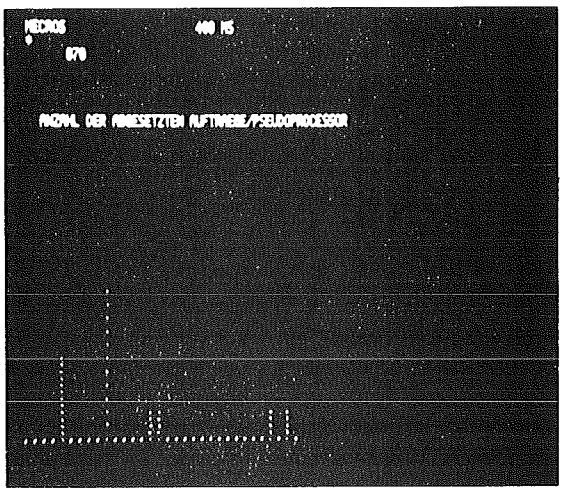


Bild 9 Auftragsliste: Anzahl der im Meßintervall abgesetzten Aufträge/Prioritätsebene

dienungskomfort möglich. Da der Aufbau von Meßroutinen jedoch ein einmaliger Vorgang ist, wurde bei diesem experimentellen System der Schwerpunkt zunächst auf den komfortablen Aufruf der Messungen gelegt.

Was dem Benutzer jedoch auch von einem noch so komfortabel gestalteten Meßsystem nicht abgenommen werden kann, ist eine detaillierte Kenntnis der Moduln des Betriebssystems, in denen er Messungen durchführen will. Hier hat sich bei dem Entwurf von Messungen gezeigt, daß es sehr vorteilhaft ist, wenn das zu messende Betriebssystem klar gegliederte Module mit gut definierten Schnittstellen enthält und alle Informationen über Systemzustände und Betriebsmittel in übersichtlichen Listen stehen. Am günstigsten ist es zweifellos, wenn bereits beim Entwurf des Betriebssystems der Einsatz eines Meßsystems durch die Bereitstellung von Meßpunkten und speziellen Meßinformationen in Meßlisten berücksichtigt wird.

Literatur:

(1) Cantrell, H.N.; Ellison, A.L.
Multiprogramming System Performance Measurement and Analysis
AFIPS Proc. 32, SJCC 1968, pp. 213-221

(2) Deniston, W.R.
SIPE: A TSS/360 Software Measurement Technique
ACM Proc. 24th Nat. Conf. August 1969, pp. 229-245

(3) Grauer, H.; Herbstreith, H.
Measurement and Evaluation of the Characteristics of a Real-Time Operating System
Third European Seminar on Real-Time Programming, Ispra, Italy, May 1973

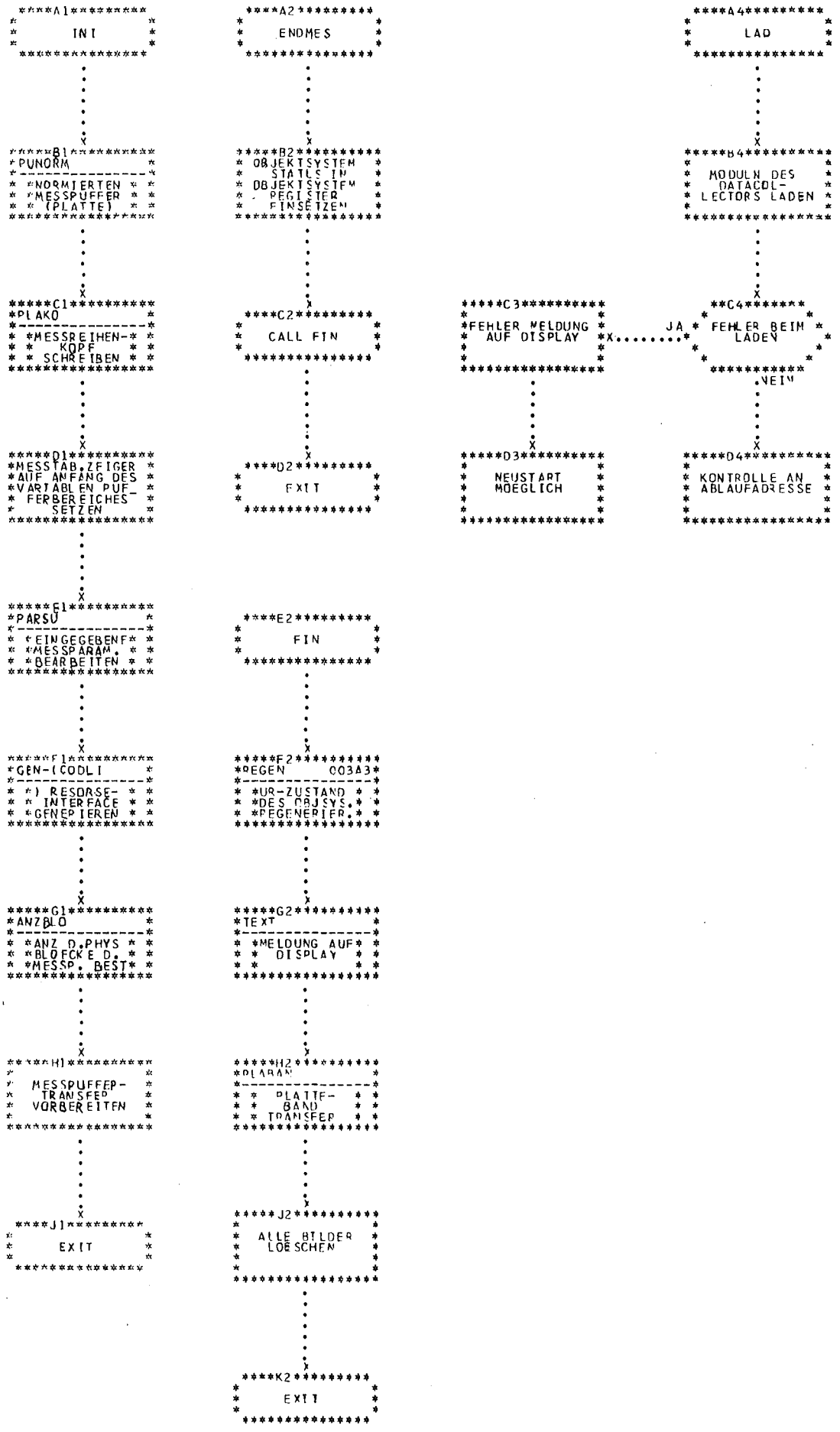
(4) Hepke, G.; Herbstreith, H.
Eingriffsorganisation für schnelle Aufgabenwechsel in einem Realzeitsystem
KFK-Bericht 1531, Januar 1973

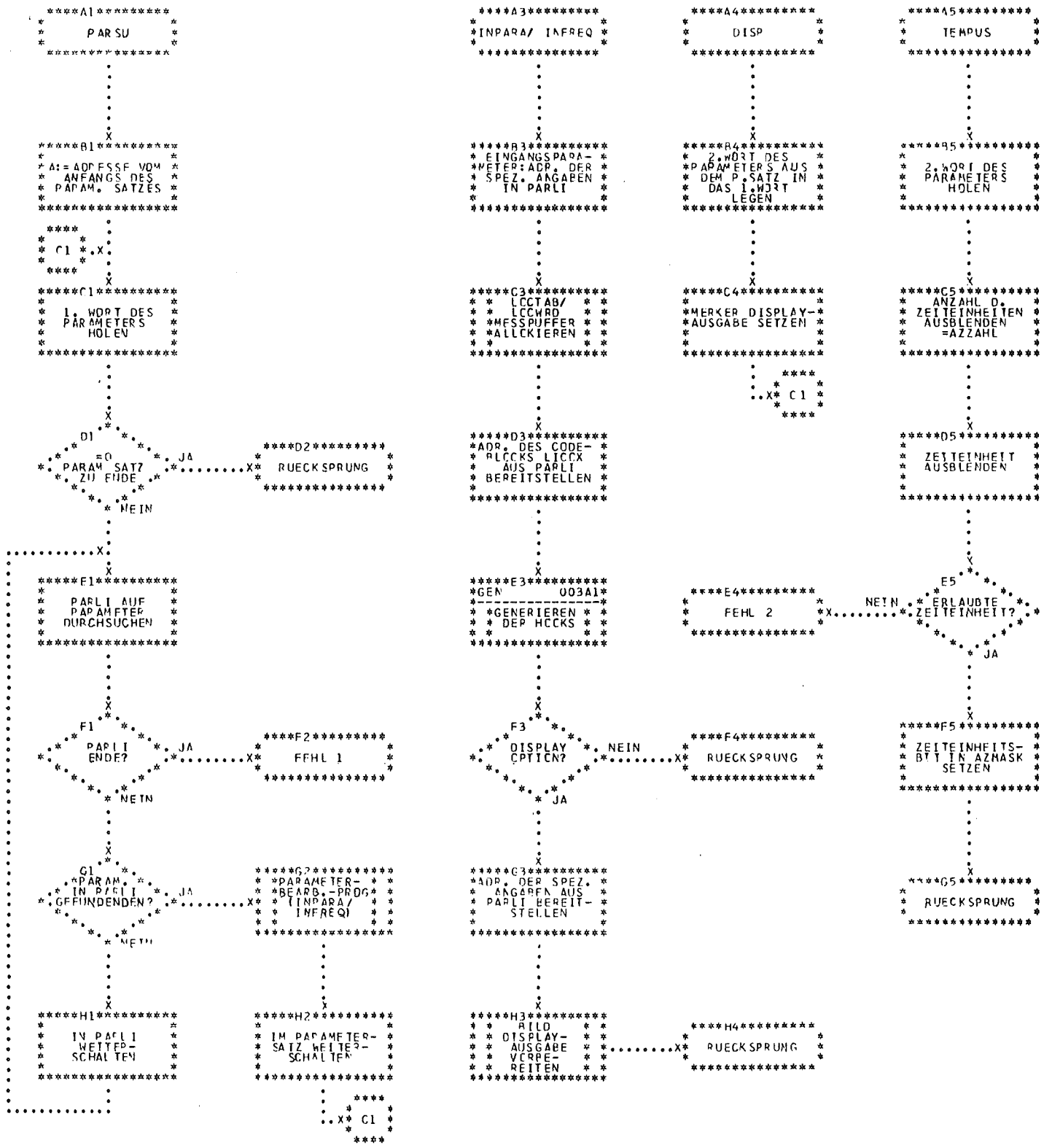
(5) Herbstreith, H.; Hepke, G.
Ablaufsteuerung für ein Realzeitsystem mit einfacher Hardwarestruktur
KFK-Bericht 1530, August 1972

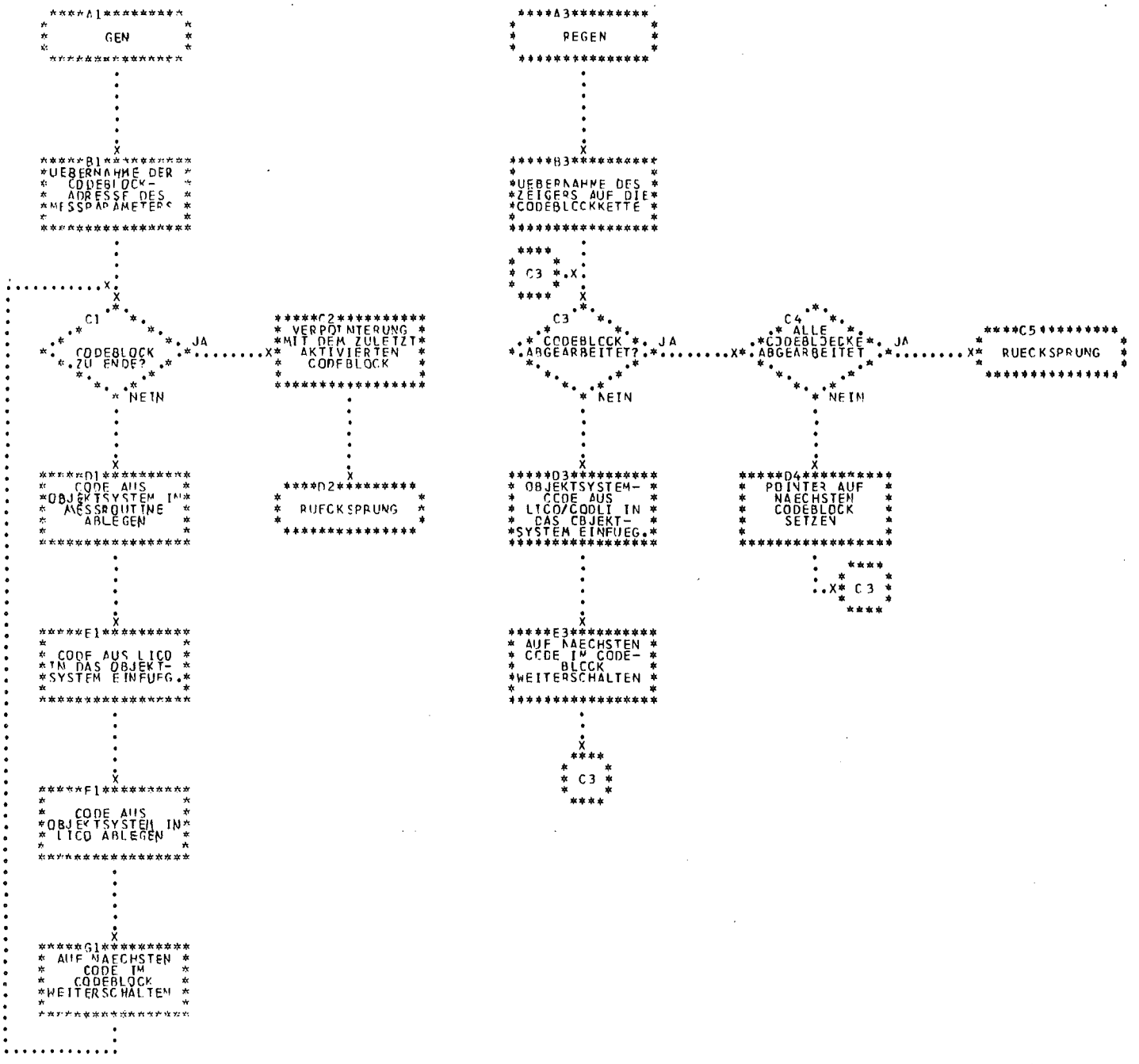
(6) Stanley, W.I.; Herkl, H.F.
Statistics Gathering and Simulation for the Apollo Real-Time Operating System
IBM Syst.Journal, Vol. 7, No. 2, 1968, pp. 299-308

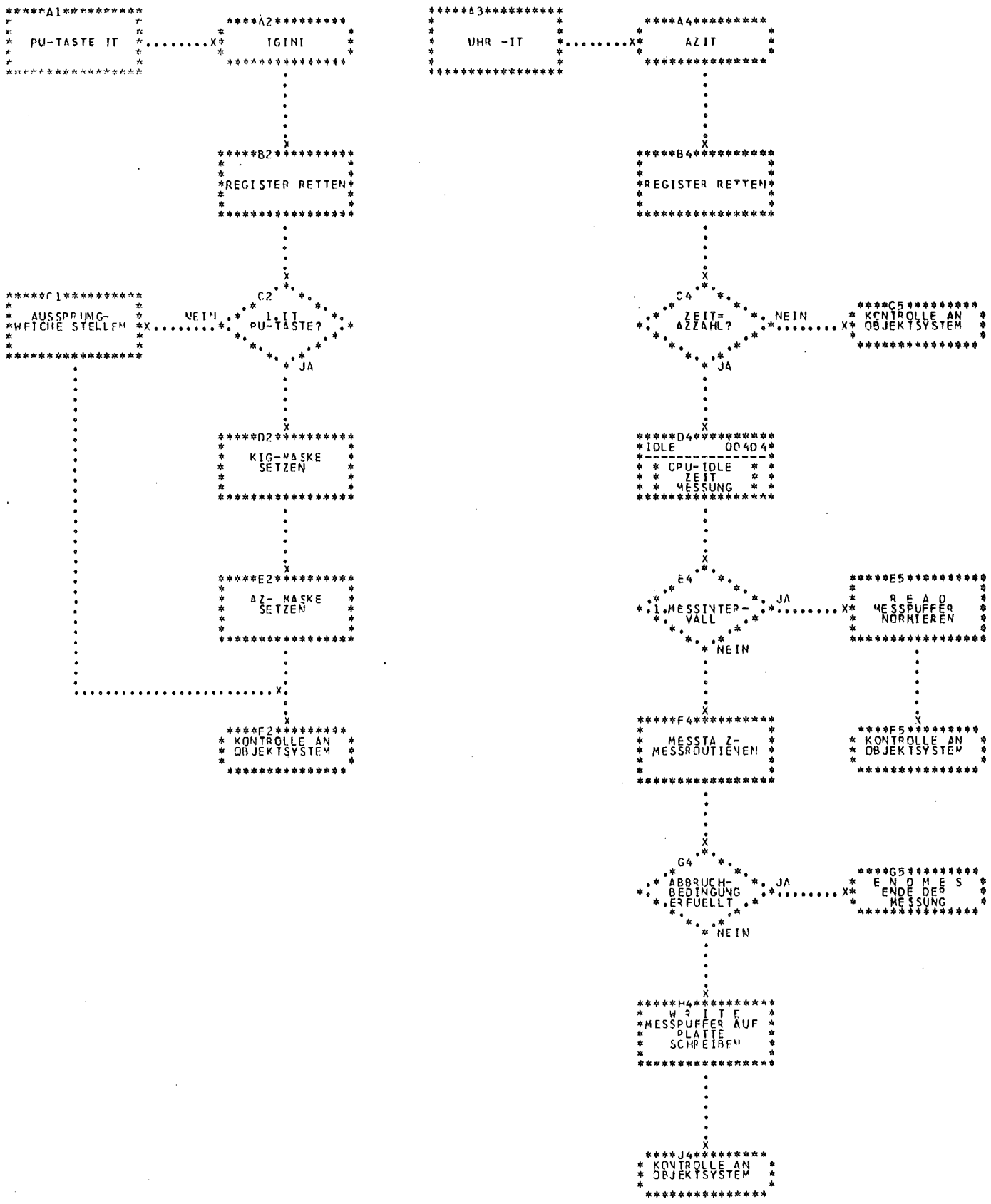
Anhang

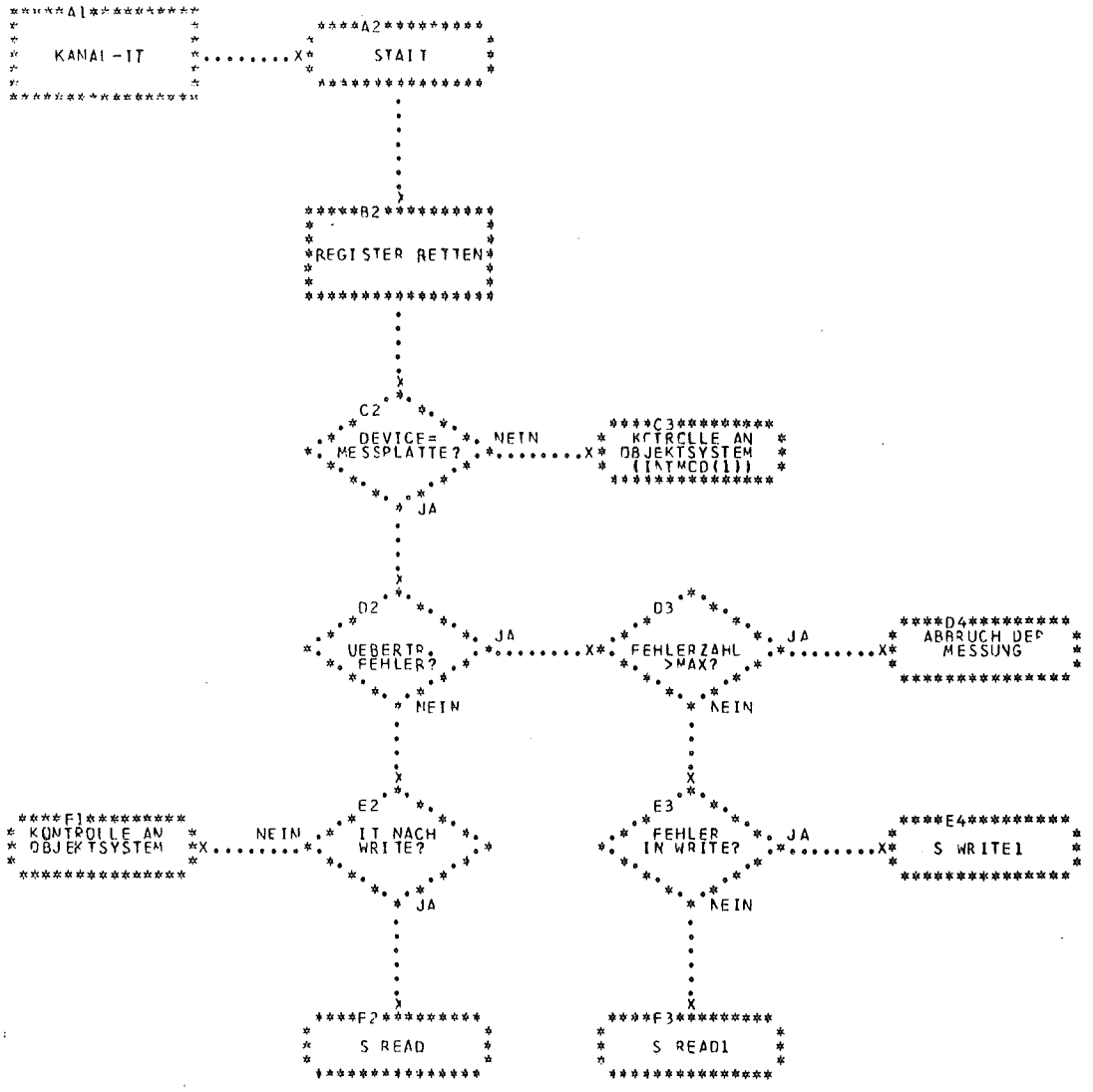
Flußdiagramme von MECROS


A1
INI
A2
ENDMES
A4
LAD
B1
PUNORM
NORMIERTEN
MESSPUFFER
(PLATTE)
B2
OBJEKTSYSTEM
STATUS IN
OBJEKTSYSTEM
REGISTER
EINSETZEN
B4
MODULN DES
DATACOL-
LECTORS LADEN
C1
PLAKO
MESSREIHEN-
KOPF
SCHREIBEN
C2
CALL FIN
C3
FEHLER MELDUNG
AUF DISPLAY
C4
FEHLER BEIM
LADEN
JA
NEIN
D1
MESSTAB.ZEIGER
AUF ANFANG DES
VARIABLEN PUF-
FERBEREICHES
SETZEN
D2
EXIT
D3
NEUSTART
MOEGLICH
D4
KONTROLLE AN
ABLAUFADRESSE
E1
PARSU
EINGEGEBENE
MESSPARAM.
BEARBEITEN
E2
FIN
F1
GEN-CODLI
) RESORSE-
INTERFACE
GENERIEREN
F2
REGEN 003A3
UR-ZUSTAND
DES OBJSYS.
REGENERIER.
G1
ANZBLO
ANZ D.PHYS
BLOECKE D.
MESSP. BEST
G2
TEXT
MELDUNG AUF
DISPLAY
H1
MESSPUFFER-
TRANSFER
VORBEREITEN
H2
PLABAN
PLATTE-
BAND
TRANSFER
J1
EXIT
J2
ALLE BILDER
LOESCHEN
K2
EXIT


A1
PARSU
B1
A1:=ADRESSE VOM ANFANGS DES PARAM. SATZES
C1
1. WORT DES PARAMETERS HOLEN
D1
=0 PARAM SATZ ZU ENDE?
JA
D2
RUECKSPRUNG
NEIN
E1
PARLI AUF PARAMETER DURCHSUCHEN
F1
PARLI ENDE?
JA
F2
FEHL 1
NEIN
G1
PARAM. IN PARLI GEFUNDENDEN?
JA
G2
PARAMETER-BEARB.-PROG (INPARA/INFREQ)
NEIN
H1
IN PARLI WEITER-SCHALTEN
H2
IM PARAMETER-SATZ WEITER-SCHALTEN
C1
A3
INPARA/ INFREQ
B3
EINGANGSPARAMETER: ADR. DER SPEZ. ANGABEN IN PARLI
C3
LCCTAB/ LCCWRD *MESSPUFFER *ALLOCKIEREN
D3
ADR. DES CODE-BLOCKS LICCX AUS PARLI BEREITSTELLEN
E3
GEN 003A1 *GENERIEREN DER HCCKS
F3
DISPLAY OPTION?
NEIN
JA
G3
ADR. DER SPEZ. ANGABEN AUS PARLI BEREIT-STELLEN
H3
ALLO DISPLAY-AUSGABE VERBE-REITEN
A4
DISP
B4
2.WORT DES PARAMETERS AUS DEM P.SATZ IN DAS 1.WORT LEGEN
C4
MERKER DISPLAY-AUSGABE SETZEN
C1
E4
FEHL 2
F4
RUECKSPRUNG
H4
RUECKSPRUNG
A5
TEMPUS
B5
2.WORT DES PARAMETERS HOLEN
C5
ANZAHL D. ZEITEINHEITEN AUSBLENDEN =AZZAHL
D5
ZEITEINHEIT AUSBLENDEN
E5
ERLAUBTE ZEITEINHEIT?
NEIN
JA
F5
ZEITEINHEITS-BIT IN AZMASK SETZEN
G5
RUECKSPRUNG


A1
GEN
B1
UEBERNAHME DER CODEBLOCK- ADRESSE DES MESSPARAMETERS
C1
CODEBLOCK ZU ENDE?
JA
NEIN
C2
VERPOINTERUNG MIT DEM ZULETZT AKTIVIERTEN CODEBLOCK
D2
RUECKSPRUNG
D1
CODE AUS OBJEKTSYSTEM IN ASSROUTINE ABLEGEN
E1
CODE AUS LICO IN DAS OBJEKT- SYSTEM EINFUEG.
F1
CODE AUS OBJEKTSYSTEM IN LICO ABLEGEN
G1
AUF NAECHSTEN CODE IM CODEBLOCK WEITERSCHALTEN
A3
REGEN
B3
UEBERNAHME DES ZEIGERS AUF DIE CODEBLOCKKETTE
C3
CODEBLOCK ABGEARBEITET?
JA
NEIN
C4
ALLE CODEBLOECKE ABGEARBEITET
JA
NEIN
C5
RUECKSPRUNG
D3
OBJEKTSYSTEM- CODE AUS LICO/CODLI IN DAS OBJEKT- SYSTEM EINFUEG.
D4
POINTER AUF NAECHSTEN CODEBLOCK SETZEN
C3
E3
AUF NAECHSTEN CODE IM CODE- BLOCK WEITERSCHALTEN
C3


A1
PU-TASTE IT
A2
IGINI
A3
UHR -IT
A4
AZIT
B2
REGISTER RETTEN
B4
REGISTER RETTEN
C1
AUSSPRUNG-WEICHE STELLEN
C2
1.IT PU-TASTE?
NEIN
JA
C4
ZEIT= AZZAHL?
NEIN
JA
C5
KONTROLLE AN OBJEKTSYSTEM
D2
KIG-MASKE SETZEN
D4
IDLE 004D4
CPU-IDLE ZEIT MESSUNG
E2
AZ- MASKE SETZEN
E4
1.MESSINTER-VALL
JA
NEIN
E5
READ MESSPUFFER NORMIEREN
F2
KONTROLLE AN OBJEKTSYSTEM
F4
MESSTA Z-MESSROUTINEN
F5
KONTROLLE AN OBJEKTSYSTEM
G4
ABBRUCH-BEDINGUNG ERFUELLT
JA
NEIN
G5
ENDMES ENDE DER MESSUNG
H4
WRITE MESSPUFFER AUF PLATTE SCHREIBEN
J4
KONTROLLE AN OBJEKTSYSTEM


A1
KANAL-IT
A2
STAIT
B2
REGISTER RETTEN
C2
DEVICE= MESSPLATTE?
NEIN
C3
KONTROLLE AN OBJEKTSYSTEM (INTMOD(1))
JA
D2
UEBERTR. FEHLER?
JA
D3
FEHLERZAHL >MAX?
JA
D4
ABBRUCH DER MESSUNG
NEIN
NEIN
E2
IT NACH WRITE?
NEIN
E3
FEHLER IN WRITE?
JA
E4
S WRITE1
F1
KONTROLLE AN OBJEKTSYSTEM
JA
NEIN
F2
S READ
F3
S READ1