KERNFORSCHUNGSZENTRUM

KARLSRUHE

August 1972 KFK 1530

Institut für Datenverarbeitung in der Technik

Ablaufsteuerung für ein Realzeitsystem mit einfacher Hardwarestruktur

H. Herbstreith
G. Hepke

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.

KARLSRUHE

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.
KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

August 1972 KFK 1530

Institut für Datenverarbeitung in der Technik

Ablaufsteuerung für ein Realzeitsystem mit einfacher Hardwarestruktur

H. Herbstreith
G. Hepke

Gesellschaft für Kernforschung m.b.H., Karlsruhe

Kurzfassung

Die Arbeiten in einem Realzeitsystem werden von weitgehend unabhängigen Prozessen erledigt. Die Steuerung der Prozeßabläufe richtet sich nach dem aktuellen Systemzustand, der von einem Zustandsvektor beschrieben wird.

Dessen Komponenten sind die Zustände der einzelnen Prozesse im System.

Will man eine für ein Realzeitsystem notwendige reaktionsschnelle Ablaufsteuerung erreichen, schränkt man das System auf wenige relevante Zustände ein.

Dies bedingt eine Einschränkung der zulässigen Prozeßzustände und der zulässigen Zustandsübergänge.

Ein allgemeines Konzept wird dargestellt am Beispiel der speziellen Implementation in CALAS70 (Computer Aided Laboratory Automation System 1970).

Abstract

Within realtime operating systems all tasks are performed by comparatively independent sequential processes. The control of the process dynamics is dependent on the actual state of the system as described by a state vector. The components of the state vector are the states of the individual system processes.

To achieve the necessary fast reaction times for a realtime system, the dispatching strategies for multiplexing the single processor among processes have to be designed in a time efficient way. Therefore the system states are reduced to a few relevant ones. A general purpose method is described at the example of a special implementation in CALAS70 (Computer Aided Laboratory Automation System 1970).

Einführung

Eine der wesentlichen Grundfunktionen eines jeden DV-Systems ist die Verwaltung des oder der zentralen Rechnerkerne. Man bezeichnet diese Funktion als Rechnerkernvergabe oder Ablaufsteuerung. Sie gehört logisch zur Aufgabe der allgemeinen Betriebsmittelvergabe eines Betriebssystems, muß aber aus nachfolgenden Gründen organisatorisch gesondert behandelt werden.

Da die Bearbeitung jeder Aufgabe im System, also auch die Betriebsmittelvergabe selbst, die Zuweisung eines Rechnerkerns voraussetzt, muß der Zuweisungsmechanismus zwangsläufig als unterste Basisfunktion ausgebildet sein. Die Rechnerkernvergabe ist deshalb, im Gegensatz zur allgemeinen Betriebsmittelvergabe, stets ein Teil der untersten Ebene der Betriebssystemhierarchie oder noch Bestandteil der Hardware.

In einfachen Realzeitsystemen, bei denen die Aufgaben als sequentielle Folge von Eingriffsprogrammen ablaufen, übernimmt das Eingriffswerk, also die Hardware, die Zuordnung des Rechnerkerns. Bei komplexeren Organisationen sind die heutigen Rechnerstrukturen noch nicht in der Lage, den Zuweisungsmechanismus auf der Hardware-Ebene abzuhandeln. Die Rechnerkernvergabe ist dann Teil des sog. Nucleus, der die unterste von drei Hierarchiestufen des Betriebssystems darstellt und als reine Erweiterung der Hardware angesehen werden kann. Neben der Rechnerkernvergabe übernimmt der Nucleus noch zwei andere exekutive Funktionen, die Eingriffsbehandlung und die Kommunikation zwischen den internen Verwaltungseinheiten des Systems.

Diese Arbeit soll das Konzept der Rechnerkernvergabe beschreiben, wie es in dem Realzeitbetriebssystem CALAS70 für das Laborautomatisierungssystem CALAS (1) implementiert wurde. Nach der Betrachtung der gesamten Betriebssystemstruktur wird detailliert auf die prioritätsgesteuerte Ablaufsteuerung eingegangen. Es wird gezeigt, wie auch unter der Voraussetzung einfacher Hardwarestrukturen, die in Realzeitsystemen notwendigen schnellen Aufgabenwechsel durch eine zeiteffiziente Ablaufsteuerung erreicht werden können.

Systemkonzept

CALAS70 wurde für einen Telefunken TR86 Rechner entworfen. Die TR86 ist eine wortorientierte Einprozessor-Maschine mit einer Wortlänge von 24 Bits. Sie besitzt ein programmierbares Register, einen Befehlssatz von 16 Grundbefehlen und einen direkt adressierbaren Kernspeicher bis zu 64 k Worten. Das Eingriffswerk, das 24 Eingriffsebenen mit festen Prioritäten unterscheidet, ist durch einen Befehl nur als Ganzes maskierbar. Das Fehlen von Index- und Basisregistern läßt die Reentrantprogrammierung, die freie Verschieblichkeit von Programmen und eine dynamische Verwaltung des Arbeitsspeichers nur umständlich und zeitraubend zu. Diese Einschränkungen machten eine weitgehende Reduktion und Anpassung bestehender Vorschläge für Betriebssysteme notwendig.

Die innere Organisation des Systems basiert auf dem von Dijkstra angegebenen allgemeinen Konzept zur hierarchischen Verwaltung von kooperativen sequentiellen Prozessen (2). Lampson (3) gab zu dem Begriff Prozeß folgende Erläuterung.

Jede natürlich logisch zusammenhängende Aktion im System, die in sich zwangsläufig sequentiell und von anderen Aktionen relativ unabhängig ist, wird als Prozeß bezeichnet.

Auf reelle Systembegriffe abgebildet bedeutet dies, daß den meisten Systemfunktionen, wie z.B. Ein- Ausgabefunktionen, der Kernspeicherverwaltung, dem Kommandoentschlüssler usw. wie auch allen Benutzerprogrammen jeweils ein Prozeß zugeordnet ist.

Darüberhinaus können auch die durch asynchrone externe oder interne Hardware-Signale ausgelösten Systemaktionen (Interruptroutinen) in den Prozeßbegriff impliziert werden.

Saltzer (4) hat in seiner allgemeinen Beschreibung von Betriebssystemen darauf hingewiesen, daß man die Einführung des Eingriffsprozesses vermeiden sollte. Man beschränkt jede direkte Eingriffsbehandlung im wesentlichen auf die Benachrichtigung eines korrespondierenden Prozesses, der die eigentliche Bearbeitung übernimmt. So wird eine einheitliche Behandlung aller Prozesse als Grundele-

mente dynamischer Abläufe mit nicht voraussehbaren Ablaufgeschwindigkeiten erreicht. Die Vorteile dieser Systematik sind:

1. Die Systemstruktur ist klar gliederbar und dadurch transparent.
2. Die ablaufenden Wechselvorgänge sind überschaubarer und besser verfolgbar.
3. Die Implementierung und der Test des Systems wird einfacher.

In Realzeitsystemen ist diese weitgehende Systematik in bestimmten Fällen nicht mehr möglich. Aufgrund der dort vorliegenden harten Reaktions- und Antwortzeitanforderungen muß ein Mechanismus existieren, der die direkte dynamische Kopplung von zeitkritischen Benutzerfunktionen an die entsprechende Eingriffsebene erlaubt. Man umgeht so den relativ zeitaufwendigen Umweg über die Benachrichtigung eines korrespondierenden Standardprozesses. Die dynamische Kopplung von Benutzerfunktionen an Eingriffsebenen wird von den Standard-Prozessen ausgelöst, wobei einer Eingriffsebene zu verschiedenen Zeitpunkten auch verschiedene Benutzerfunktionen zugeordnet sein können. Die Dynamik dieser Vorgänge und der mögliche Analogievergleich mit den Standard-Prozessen rechtfertigt die Einführung der Eingriffsprozesse als zweites Grundelement dynamischer Abläufe im Betriebssystem.

Demnach besteht das gesamte Betriebssystem aus einer Menge von Standard-Prozessen, Eingriffsprozessen und einem Basissystem, das diese beiden Verwaltungseinheiten organisiert und verwaltet.

Prozeßbearbeitung mittels Pseudoprozessoren

Definitionsgemäß sind die Prozesse selbständige und relativ unabhängige Einheiten, wodurch ihre simultane Bearbeitung möglich wird. Dazu stellt das Betriebssystem jedem existierenden Prozeß einen sog. Pseudo-Prozessor zur Verfügung, der die Bearbeitung nach Zuweisung des realen Prozessors übernimmt.

Durch die organisatorische Trennung der Prozesse in Eingriffs- und Standardprozesse müssen zwei Typen von Pseudoprozessoren

unterschieden werden, Standardpseudoprozessoren und Eingriffs-pseudoprozessoren.

Standardpseudoprozessoren

Im folgenden wird die Struktur und Organisation der Standard-pseudoprozessoren beschrieben, wobei zur Vereinfachung das Vorwort Standard weggelassen ist.

Um die Zuweisung des realen Prozessors ohne großen Adaptionsaufwand durchzuführen, ist die Struktur der Pseudoprozessoren, so weit als möglich, dem realen Prozessor angepaßt. So besitzen sie einen Befehlszähler, Arbeitsregister, einen speziellen Befehlssatz und zusätzlich einen lokalen Registersatz zur Aufnahme von Zustands- und Beschreibungsgrößen. Die Inhalte aller Register bilden zusammen den aktuellen internen Statusvektor eines Pseudoprozessors. Neben diesem internen Statusvektor wird jeder Pseudoprozessor durch einen externen Statusvektor repräsentiert. In ihm ist alle Information über seine aktuelle Disposition enthalten, die für die Vergabe des realen Prozessors relevant ist.

Jeder Pseudoprozessor ist also durch die Angabe der beiden Statusvektoren und des speziellen Befehlssatzes dynamisch und statisch vollständig beschrieben.

Wie schon erwähnt, muß das Betriebssystem jedem existierenden Prozeß einen durch die beiden Zustandsvektoren beschriebenen Pseudoprozessor verfügbar machen. Dies bedeutet aber nicht, daß genau so viele Pseudoprozessoren wie Prozesse existieren müssen. Die Anzahl der notwendigen Pseudoprozessoren wird ausschließlich bestimmt durch die Anzahl der Prozesse, die natürlich simultan bearbeitbar sind. Nicht simultan bearbeitbar sind nun solche Prozesse, die gemeinsam sequentielle Betriebsmittel besitzen. Beispiele dafür sind die E/A-Prozesse eines Kanals oder eines Geräts, aber auch alle Prozesse, die mit gemeinsamen Prozeduren arbeiten, da das vorliegende System keine Reentrantprozeduren kennt. Alle diese Prozesse, die sich gegenseitig nicht überholen können, werden einem gemeinsamen Pseudoprozessor zugeordnet. Jeder Pseudoprozessor besitzt deshalb eine Warteschlange, in der

die zur Bearbeitung anstehenden Prozesse gemäß ihrer Priorität eingereiht sind. Durch diese eingeschränkte Simultanverarbeitung von Prozessen reduziert sich die Anzahl der notwendigen Pseudoprozessoren soweit, daß eine Festlegung auf eine feste Anzahl bereits bei der Systemgenerierung sinnvoll wird. Man verzichtet also auf ihre dynamische Verwaltung, was den Verwaltungsaufwand stark reduziert. In CALAS7O sind 4 Funktionsklassen von Pseudoprozessoren unterschieden zur Bearbeitung der Prozesse für:

1. Systemdienste (Ein- Ausgabe, Kernspeicherverwaltung, Zeitdienst usw.)
2. Mensch-Maschine-Kommunikation
3. Realzeitaufgaben
4. nichtzeitkritische Hintergrundaufgaben

Der interne Statusvektor eines jeden Pseudoprozessors dieser 4 Klassen besteht aus:

1. Name
2. Prioritätsnummer
3. Befehlszähler
4. Arbeitsregister
5. Beschreibung des Adressenraumes der dem gerade bearbeiteten Prozeß zugeordneten Prozeduren und Daten
6. Auflistung aller an andere Pseudoprozessoren abgegebenen Prozesse
7. Auflistung aller Synchronisationsbedingungen
8. Auflistung aller temporär zugeordneten Betriebsmittel
9. aktuelle und maximale Länge der Warteschlange hinter dem Pseudoprozessor
10. Ablage des Vorwärtszeigers zur Warteschlange bei blockiertem Pseudoprozessor

Der externe Statusvektor eines Pseudoprozessors kann drei Grundzustände annehmen:

1. bereit, d.h. der Pseudoprozessor wartet nur auf die Zuweisung des realen Prozessors
2. blockiert, d.h. er wartet auf eine Synchronisation durch einen anderen Pseudoprozessor oder es ist ihm zur Zeit kein Prozeß zugeordnet
3. aktiv, d.h. der reale Prozessor ist dem Pseudoprozessor gerade zugeordnet

Strukturell unterscheiden sich die Pseudoprozessoren nur in ihren speziellen Befehlssätzen. Je nach Art der zu bearbeitenden Prozesse sind einige der reellen Hardwarebefehle verboten. Benutzerprozessoren kennen z.B. keine E/A-Befehle oder sonstige den Systemprozessoren vorbehaltene Instruktionen.

Dagegen besitzen alle Pseudoprozessoren sog. Pseudobefehle, die in der tatsächlichen Hardware nicht vorhanden und deshalb durch Basisfunktionen im Nucleus des Betriebssystems in Software nachgebildet sind. Sie laufen analog zu den realen Befehlen unterbrechungsgeschützt ab. Funktionell sind in CALAS7O drei Typen von Pseudobefehlen unterschieden und implementiert.

1. Pseudobefehle zur dynamischen Verwaltung der Prozesse

   call — generiert einen neuen Prozeß und ordnet ihn in die Warteschlange eines Pseudoprozessors ein. Die Parameter-Versorgung erfolgt über eine Referenzadresse im Arbeitsregister.

   exit — vernichtet einen Prozeß, wenn keine Abhängigkeit mit anderen Prozessen besteht. Der Prozeß wird aus der entsprechenden Warteschlange entfernt und alle belegten Betriebsmittel freigegeben, d.h. der Statusvektor wird teilweise normiert.

pass generiert einen neuen Prozeß und vernichtet danach den Prozeß, der den Pseudobefehl ausgelöst hat. Dabei werden alle Synchronisationsbedingungen weitergegeben.

2. Pseudobefehle zur Kommunikation zwischen Prozessen

sleep versetzt einen Pseudoprozessor in den blockierten Zustand, wenn die Synchronisationsbedingungen noch nicht erfüllt sind. Die Synchronisationsbedingung muß als Parameter mitgegeben werden.

wakeup setzt einen blockierten Pseudoprozessor wieder in den bereiten Zustand.

3. Ein Pseudobefehl zum dynamischen Aufruf von Eingriffsprozessen

callit generiert einen Eingriffsprozeß, der beim Auftreten des Eingriffs abläuft und gegebenenfalls den Pseudoprozessor benachrichtigt, der den Pseudobefehl ausgeführt hat.

Eingriffspseudoprozessoren

Die Eingriffspseudoprozessoren übernehmen die Bearbeitung der sog. Eingriffsprozesse, die sich in drei wesentlichen Punkten von den Standardprozessen unterscheiden.

1. Die Aktivierung der Eingriffsprozesse, d.h. die Zuweisung des realen Prozessors erfolgt ausschließlich durch das Eingriffswerk, und die zentrale Eingriffsbehandlung im Nucleus des Betriebssystems anhand der hardwaremäßig vorgegebenen Prioritäten.
2. Die Eingriffsprozesse laufen vollständig unterbrechungsgeschützt und deshalb starr sequentiell ab.

3. Es besteht vollständige Unabhängigkeit zwischen den Eingriffsprozessen, d.h. eine Kommunikation unter ihnen ist nicht möglich.

Bedingt durch diese spezifischen Eigenschaften der Eingriffsprozesse ergibt sich eine sehr einfache Struktur der Eingriffspseudoprozessoren, die wie die der Standardpseudoprozessoren durch die Angabe der beiden Statusvektoren und des speziellen Befehlssatzes beschrieben ist. Im Gegensatz zu den Standardpseudoprozessoren fehlen die Arbeitsregister und die zur Synchronisation notwendigen Statusregister.

Der interne Statusvektor besteht aus:

1. Nummer
2. Priorität
3. Beschreibung des Adressenraumes des Eingriffsprozesses
4. Identifikation des korrespondierenden Standardprozesses

Der externe Statusvektor der Eingriffspseudoprozessoren kann in Analogie zu den Standardpseudoprozessoren ebenfalls drei Grundzustände annehmen.

1. bereit, d.h. der Pseudoprozessor wartet nur auf die Zuweisung des realen Prozessors durch das Auftreten des Eingriffsignals.
2. blockiert, d.h. die zugehörige Eingriffsebene ist maskiert.
3. aktiv, d.h. der Eingriff ist angenommen und der reale Prozessor ist dem Eingriffspseudoprozessor zugeordnet.

Zur Synchronisation mit Standardprozessen und zur dynamischen Verwaltung der Eingriffsprozesse ist der Befehlssatz der Eingriffspseudoprozessoren um drei Pseudobefehle erweitert. Sie haben folgende Charakteristik:

call — generiert einen Standardprozess (siehe Standardpseudoprozessoren)

exitit vernichtet einen Eingriffsprozeß und blockiert die zugehörige Eingriffsebene.

wakeit setzt den blockierten Standardpseudoprozessor wieder in den bereiten Zustand, der den Eingriffsprozeß kreiert uns sich mit sleep auf ihn synchronisiert hat.

Eine detaillierte Beschreibung der Eingriffsorganisation ist in (5) enthalten.

Rechnerkernvergabe

Mit der Implementierung der Standardpseudoprozessoren und der Synchronisations- und Verwaltungsfunktionen durch Pseudobefehle sind fast alle wesentlichen Hilfsmittel zur Disposition und Ausführung des Bearbeitungsablaufes der sequentiellen Prozesse gegeben. Das gesamte System gleicht jetzt einem dedizierten Prozessorsystem, bei dem jedem Prozeß ein eigener Prozessor zur Bearbeitung zur Verfügung steht. Das noch verbleibende Problem ist die aktive Vergabe des einen realen Rechnerkerns an die Pseudoprozessoren. Diese exekutive Basisfunktion, Rechnerkernvergabe genannt, ist, wie in der Einleitung begründet, Teil der untersten Hierarchiestufe des Betriebssystems. Sie tritt immer dann in Aktion, wenn der Systemstatus sich in einer Weise geändert hat, daß ein Aufgabenwechsel, d.h. die Neuvergabe des Rechnerkerns erforderlich wird. Da die Rechnerkernvergabe als Basisfunktion großen Einfluß auf die Systemantwortzeiten hat, muß sie zeiteffizient entworfen und implementiert werden. Dieser Forderung kann nur genügt werden, wenn die in der Aufgabe notwendigen Suchvorgänge minimiert oder durch geschickte Anordnung der Suchobjekte beschleunigt werden können.

Im Hinblick auf die Rechnerkernvergabe ergibt sich der Systemzustand durch die Zustände der Pseudoprozessoren, d.h. durch die Menge ihrer externen Statusvektoren. Zur Erkennung und Verarbeitung von Zustandsübergängen sind die externen Statusvektoren, zusammen mit einer Warteschlangenorganisation für die den Pseudoprozessoren zugeordneten Prozesse, in einer zentralen

Liste des Nukleus, der Prioritätsliste der Pseudoprozessoren, zusammengefaßt. Zustandsübergänge werden entweder durch externe Unterbrechungen oder durch die Ausführung der Pseudobefehle verursacht und führen zu Veränderungen in dieser zentralen Liste. Bild 1 zeigt schematisch die möglichen Zustandsübergänge und ihre Auslösung.

Zur Ausführung der Rechnerkernvergabe wird notwendigerweise der einzige Rechnerkern des Systems benötigt, so daß sich die Pseudoprozessoren während dieser Zeit nur in den Zuständen "Bereit" oder "Blockiert" befinden können. Diese Tatsache vereinfacht die Struktur der in Bild 2 gezeigten Prioritätsliste der Pseudoprozessoren. Sie ist vom Typ "verzeigert" und linear aufsteigend nach den Prioritätsnummern der Pseudoprozessoren geordnet. Dabei bedeuten steigende Prioritätsnummern fehlende Priorität. Aus programmtechnischen Gründen wurden nur ganzzahlige Prioritätsnummern gewählt, da jedem Pseudoprozessor genau 2 Worte der Liste mit folgendem Inhalt zugeordnet sind:

1. Eine Zustandsflagge, die entweder "Bereit" oder "Blockiert" signalisiert.
2. Im Zustand "Bereit" je einen Vorwärts- und Rückwärtszeiger zu der Warteschlange der zugeordneten Prozesse, die ihrerseits durch doppelt verzeigerte sog. dynamische Prozeßblöcke repräsentiert werden (Bild 3).
3. Im Zustand "Blockiert" einen Vorwärtszeiger zum ersten Wort des Pseudoprozessors mit der nächst höheren Prioritätsnummer. Der Vorwärtspointer zu der Warteschlange der Prozesse wird für die Dauer der Blockierung im internen Statusvektor zwischengespeichert.

Die so strukturierte Prioritätsliste, die internen Zustandsvektoren und ein zentrales Prioritätsregister bilden die Datenbasis für die Rechnerkernvergabe, die nach dem Aufruf zeitlich nacheinander folgende drei Aufgaben durchführt:

1. Ermittlung desjenigen Pseudoprozessors, der zur Zeit die höchste Priorität hat und sich im Zustand "Bereit" befindet.
2. Allokation des Rechnerkerns durch Kopieren der Registerinhalte des ausgewählten Pseudoprozessors in die Register des realen Rechnerkerns und Übernahme der Priorität in das zentrale Prioritätsregister.
3. Start des Pseudoprozessors unter Aufhebung der Unterbrechungssperre.

Die beiden letzten Teilaufgaben sind einfacher Natur und verstehen sich ohne nähere Beschreibung, während die Selektion des zur Zeit wichtigsten Pseudoprozessors relativ komplex erscheint.

Die einfachste Lösung, die sich aus der Datenstruktur der Prioritätsliste direkt anbietet, ist ein systematisches zyklisches Abfragen dieser Liste mit Hilfe der Verzeigerung und der Zustandsflagge "Bereit" oder "Blockiert" als Kriterium. Man erreicht eine optimale Implementierung des Abfragemechanismus durch die direkte Verwendung des Indirektbits als Zustandsflagge, wodurch pro Suchschritt nur jeweils ein Speicherzugriff notwendig ist.

Leider werden die Vorteile der Systematik dieser Lösung durch die immer noch relativ große Durchlaufzeit und die zwangsläufig durch die Hardware bedingte Nichtunterbrechbarkeit des indirekten Adressierungsvorganges stark überkompensiert. So sind z.B. bei 100 implementierten Pseudoprozessoren mindestens 100 eingriffsgeschützte Speicherzyklen notwendig, um den Pseudoprozessor mit der niedrigsten Priorität zu finden.

Eine wesentliche Reduzierung dieses Zeitaufwandes kann nur unter Aufgabe der Systematik erzielt werden. Untersucht man die möglichen Ursachen der Auslösung der Rechnerkernvergabe, so stellt man fest, daß nur die Ausführung eines Pseudobefehls innerhalb der Eingriffs- und Standardprozesse zu Statusänderungen der Standardpseudoprozessoren führen kann. Man entwickelt deshalb speziell für jeden Pseudobefehl eine eigene optimale Strategie.

Dabei wird wie folgt vorgegangen:

Jeder Pseudobefehl übergibt beim Aufruf der Rechnerkernvergabe zwei Parameter, die Priorität und den neuen Status des Pseudoprozessors, auf den der Pseudobefehl gewirkt hat. Die Rechnerkernvergabe kann aus diesen beiden Parametern und dem zentralen Prioritätsregister, das die Priorität desjenigen Pseudoprozessors enthält, der zuletzt den Rechnerkern besessen hat, den Ablauf entscheiden.

Die formale Beschreibung des Entscheidungsalgorithmus könnte folgendermaßen aussehen:

```
rechnerkernvergabe: procedure(prio_neu,status);

if
  ((status='blockiert') and (prio_neu≠prio_alt)) or
  ((status='bereit') and (prio_neu ≤ prio_alt));

then do;
  call allocation; goto start;
  end;

if (status='bereit') and (prio_neu > prio_alt);

then do;

  prio_alt=prio_neu; call allocation; goto start;
  end;

if (status='blockiert') and (prio_neu=prio_alt);

then do;

  call such vorgang; call allocation; goto start;
  end;

end; rechnerkernvergabe
```

Man erkennt in der Prozedur vier unterscheidbare Fälle. In den ersten drei Bedingungssituationen kann die Rechnerkernallokation und der Start des Pseudoprozessors nach einfachen logischen Entscheidungen ohne Suchvorgänge direkt erfolgen. Nur in einem Fall,

in dem sich der Pseudoprozessor, der den Pseudobefehl ausgeführt hat, selbst blockiert und keine Statusänderung eines anderen verursacht hat, muß ein Suchvorgang angestoßen werden. Mit dem Ergebnis des Suchvorgangs, der Priorität eines breiten Pseudoprozessors, kann dann die Allokation des Rechnerkerns und der Start des Pseudoprozessors in der gleichen Weise wie in den anderen Fällen erfolgen.

Schlußbemerkungen

Beurteilt man den angegebenen Algorithmus, so ergibt sich, daß er der eingangs erwähnten Forderung nach zeitlicher Effizienz genügt, ohne wesentlich an Transparenz zu verlieren. Die in der Implementierung auf dem TR86-Rechner erzielte Ausführungszeit der Rechnerkernvergabe ohne eventuelle Suchvorgänge beträgt ~30 µsec (30 Speicherzyklen). Es ist somit gelungen, einen wesentlichen Teil der Ausführung von Aufgabenwechseln zeitlich zu optimieren, was sich indirekt auf die Antwortzeiten des Gesamtsystems günstig auswirkt.

Literaturverzeichnis

(1) Gagel, G. et al.
CALAS68 - ein computergestütztes Vielfachzugriffs-system zur Laborautomatisierung
KFK-Ext. 19/69-1, November 1970

(2) Dijkstra, E.W.
Cooperating Sequential Processes
Technological University, Eindhoven 1965

(3) Lampson, B.W.
A Scheduling Philosophy for Multiprocessing Systems
Communications of the ACM, Vol.11, No.5, May 1968

(4) Saltzer, J.H.
Traffic Control in a Multiplexed Computer System
MAC-TR-30 (THESIS), MIT, July 1966

(5) Hepke, G., Herbstreith, H.
Eingriffsorganisation für schnelle Aufgabenwechsel in einem Realzeitsystem
Bericht KFK 1531, September 1972

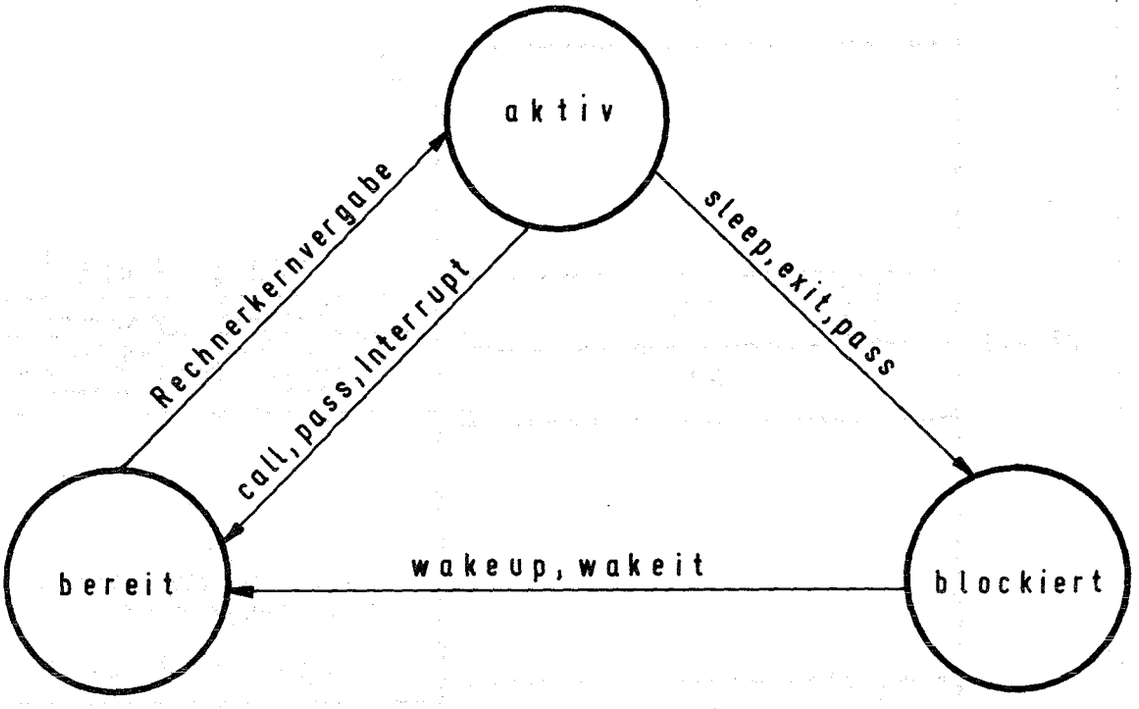
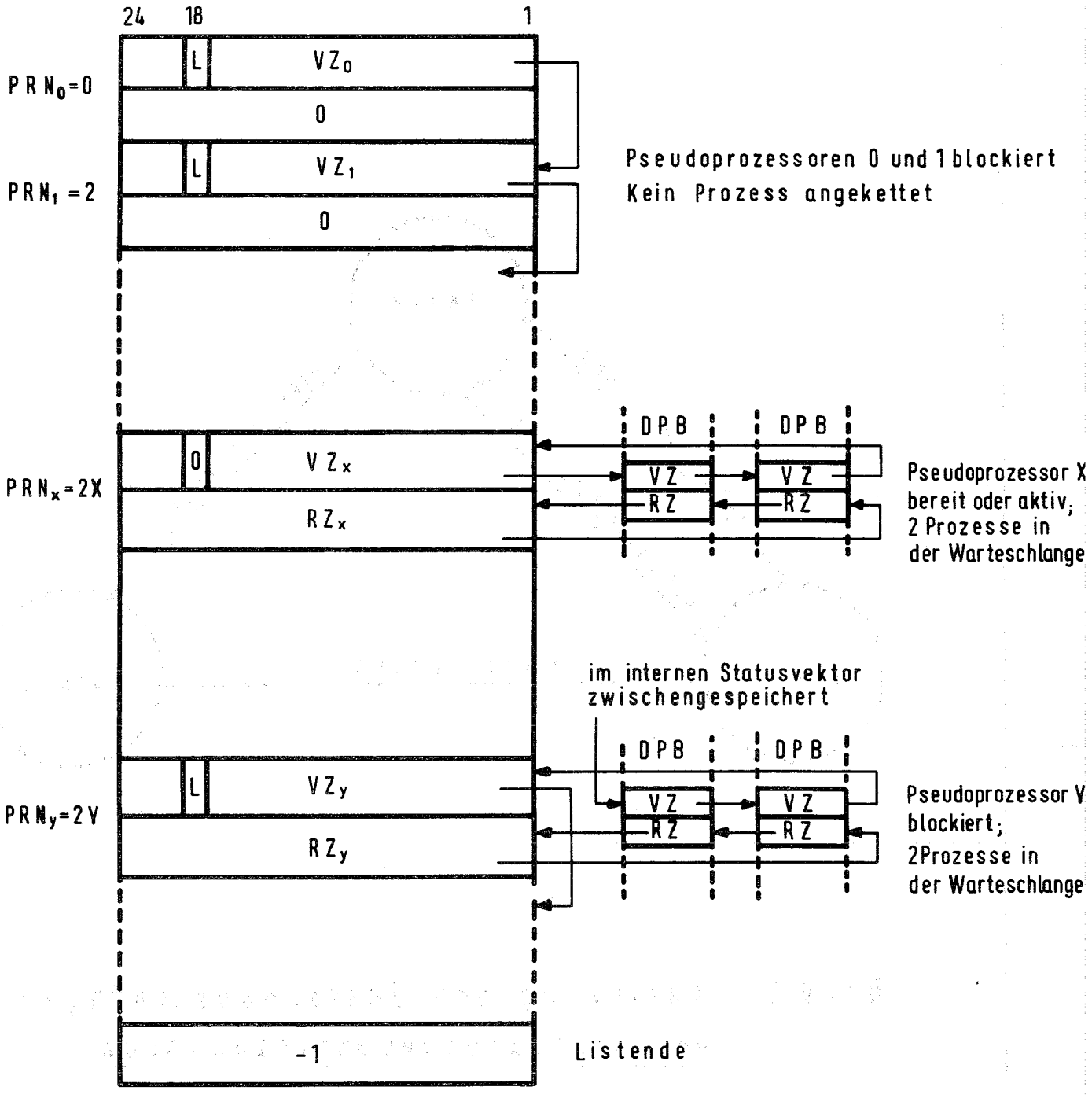
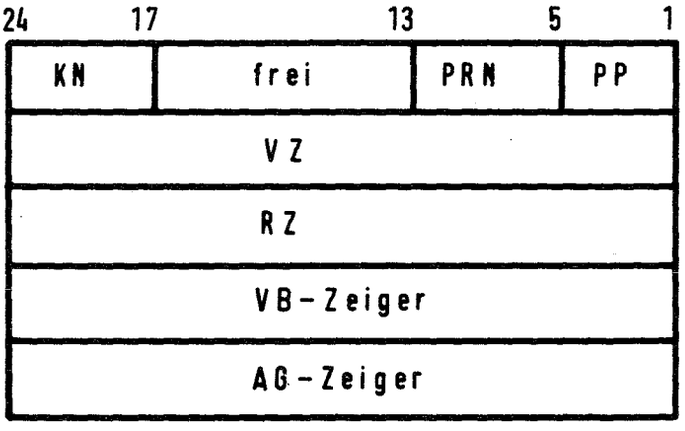


Bild 1 Auslösung von Zustandsübergängen der Standardpseudoprozessoren



PRN$_X$ = Prioritätsnummer des Standardpseudoprozessors X
VZ$_X$ = Vorwärtszeiger des Standardpseudoprozessors X
RZ$_X$ = Rückwärtszeiger des Standardpseudoprozessors X
DPB = dyn. Prozeßblöcke (siehe Bild 3)

Bild 2 Prioritätsliste der Standardpseudoprozessoren



KN = Blockkennung

PP = Prozeßspezifische Priorität

PRN = Prioritätsnummer des Standardpseudo-prozessors, an der der Prozess angemeldet werden soll

VB-Zeiger = Zeiger zum prozeßspezifischen Versorgungsblock

AG-Zeiger = Zeiger zum internen Statusvektor des Standardpseudoprozessors, der den Prozess angemeldet hat (Auftraggeberzeiger)

Bild 3 Aufbau der dyn. Prozeßblöcke (DPB)