5.1 Begriffsbestimmung

Was ist ein Betriebssystem?

Definition Betriebssystem (DIN 44300):

sind die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die Grundlage der möglichen Betriebsarten des digitalen Rechensystems bilden und insbesondere die Abwicklung von Programmen steuern und überwachen

Betriebssystem

- Systematisch aufgebaute Sammlung von Steuerungs- und Hilfsprogrammen
- Zuteilung der vorhandenen Betriebsmittel zu den konkurrierenden Rechenprozessen

Scheduling

- Erleichterung der Bedienung und Programmierung des Rechners und der angeschlossenen Geräte für den Anwender

Treiber

Eigenschaften von Betriebssystemen

- Realisierung der hardwareabhängigen Aufgaben
- häufig Mitlieferung vom Hersteller des Rechners
  - effizientes Betriebssystem setzt genaue Kenntnis der Hardwarestruktur voraus
  - vielfach für ganze Rechnerlinie
  - Amortisierung der hohen Entwicklungskosten eines Betriebssystems
- Größe
  - ⇨ **mehrere Kilo-Bytes bei Mikrorechnern**
  - ⇨ **mehrere Mega-Bytes bei Großrechnern**
- Integration klassischer Betriebssystembestandteile in Form von Halbleiterchips

Betriebsmittel

- Objekte, die die Rechenprozesse während des Ablaufs benötigen und auf deren Zuteilung warten müssen
- Geräteeinheiten
  - Prozessoren
  - Speicher
  - Peripheriegeräte wie Drucker
- Systemprogramme

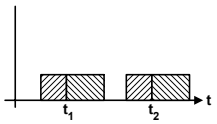
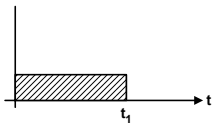
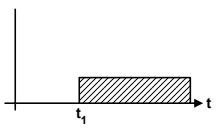
Anforderungen an ein Echtzeitbetriebssystem

Zeitbedingungen bei der Abwicklung von Rechenprozessen

Abschluss bzw. Anstoß von Rechenprozessen

- zu einem exakten Zeitpunkt
- zu einem bestimmten Zeitpunkt, aber mit zulässigen Zeittoleranzen
- zu einem spätesten Zeitpunkt
- zu einem frühesten Zeitpunkt

Beispiele von Zeitbedingungen für die Abwicklung von Rechen-prozessen

Anforderung	Zeitdiagramm	Beispiel
feste Zeitpunkte		Kennfeldauf-nahme an Prüfständen
Zeitpunkte mit Toleranzen		Erfassung von Regelgrößen bei der direkten digitalen Regelung
spätester Zeitpunkt		Erfassung von Datentelegrammen bei Linienleiter-Systemen
frühester Zeitpunkt		Folgesteuerung bei Chargen-Prozessen

Kategorien von Echtzeit-Betriebssystemen (1)

- **Echtzeit-UNIX**
  - Kompatibel zu UNIX-System V
  - Einsatz bei Prozessleitsystemen
- **Echtzeit-Kerne**
  - UNIX-kompatibler Mikro-Kernel mit
    Speicherverwaltung,
    Interruptverarbeitung,
    Scheduler,
    Taskverwaltung,
    Schnittstellen auf der Basis von TCP/IP
  - optimal an Anforderungen angepasst
  - gut optimierter Code für unterschiedliche Plattformen

Kategorien von Echtzeit-Betriebssystemen (2)

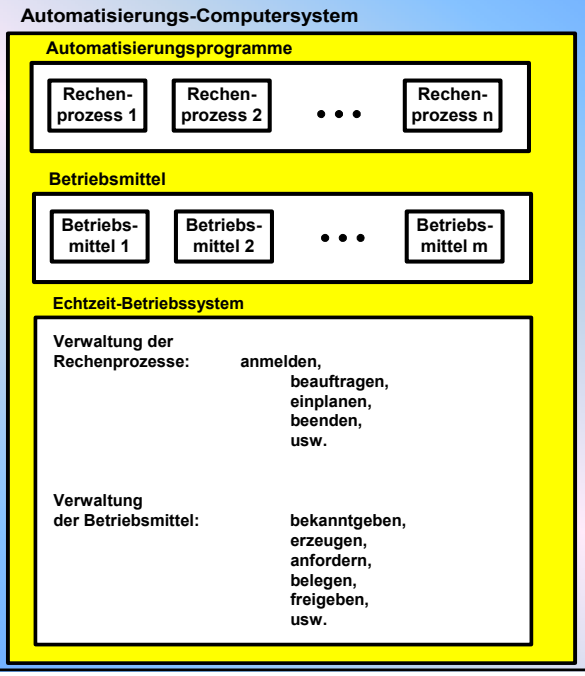
- **Echtzeit-Betriebssystemerweiterungen**
  - Erweiterung von MS-DOS-Systemen
  - Bibliothek zur Einhaltung der Echtzeitbedingungen
- **Echtzeit-Betriebssysteme**
  - sehr effizient
  - flexibel konfigurierbar
  - an UNIX orientiert

5.2 Organisationsaufgaben eines Echtzeit-Betriebssystems

Aufbau und Eigenschaften von Echtzeit-Betriebssystemen

Aufgaben eines Echtzeit-Betriebssystemen

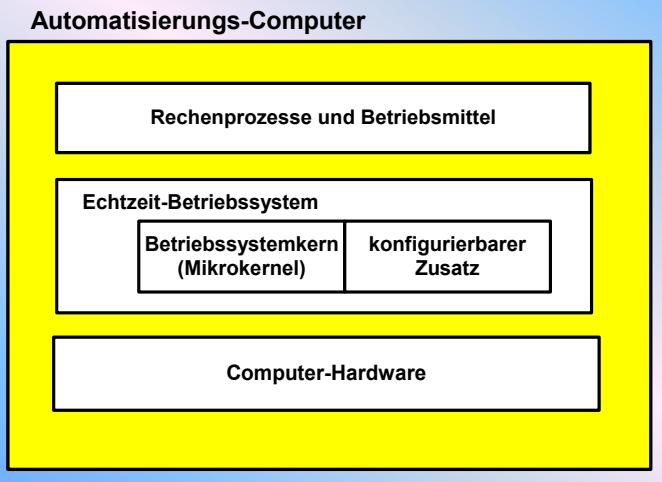
Verwaltung von Rechenprozessen und Betriebsmitteln unter Erfüllung der Forderungen nach Rechtzeitigkeit, Gleichzeitigkeit und Effizienz



Betriebssystemfunktionen

- Organisation des Ablaufs der Rechenprozesse (Scheduling)
- Organisation der Interruptverwaltung
- Organisation der Speicherverwaltung
- Organisation der Ein-/Ausgabe
- Organisation des Ablaufs bei irregulären Betriebszuständen und des (Wieder-) Anlaufs

Schichtenarchitektur eines Automatisierungs-Computersystems



Rechenprozess-Verwaltung

Arten von Rechenprozessen

- ❒ Anwenderprozesse
- ❒ Systemprozesse

⇨ **zentrale Protokollierung**

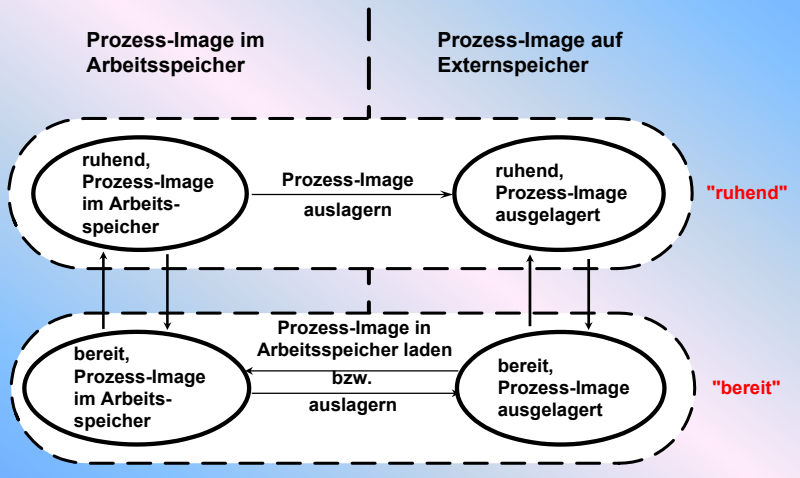
⇨ **Verwaltung von Speichermedien**

⇨ **Null-Prozess**

Aufgaben bei der Rechenprozess-Verwaltung

- ❒ Koordinierung des Ablaufs von Anwender-und System-prozessen
- ❒ Parallelbetrieb möglichst vieler Betriebsmittel
- ❒ Abarbeitung von Warteschlangen bei Betriebsmitteln
- ❒ Synchronisierung von Anwender-Systemprozessen
- ❒ Vermeidung, Entdeckung und Behebung von Deadlocks

Beispiel für zusätzliche hardwareabhängige Rechenprozess-Zustände bei einem Rechner mit Externspeicher



Interrupt-Verwaltung

- Unterbrechung des geplanten Programmablaufs
- Beauftragung einer Behandlungsroutine

Interrupt Service Routine

Speicherverwaltung

Preis von Speicherplatz proportional zur Zugriffsgeschwindigkeit

⇨ **optimale Nutzung notwendig**

Speicherhierarchie-Ebene

- Cache-Speicher
  (besonders schneller Halbleiterspeicher)
- Arbeitsspeicher
- Plattenspeicher
- Diskette

Aufgaben der Speicherverwaltung

- Optimale Ausnutzung der “schnellen” Speicher
- Koordinierung des gemeinsamen Zugriffs auf einen Speicher-bereich
- Schutz des Speicherbereichs verschiedener Rechenprozesse gegen Fehlzugriffe
- Zuweisung von physikalischen Speicheradressen für die logischen Namen in Anwenderprogrammen

Zuordnung Name im Programm zu physikalischer Adresse im Hauptspeicher

Name

↓ Compiler

Adresse relativ zum Programmanfang

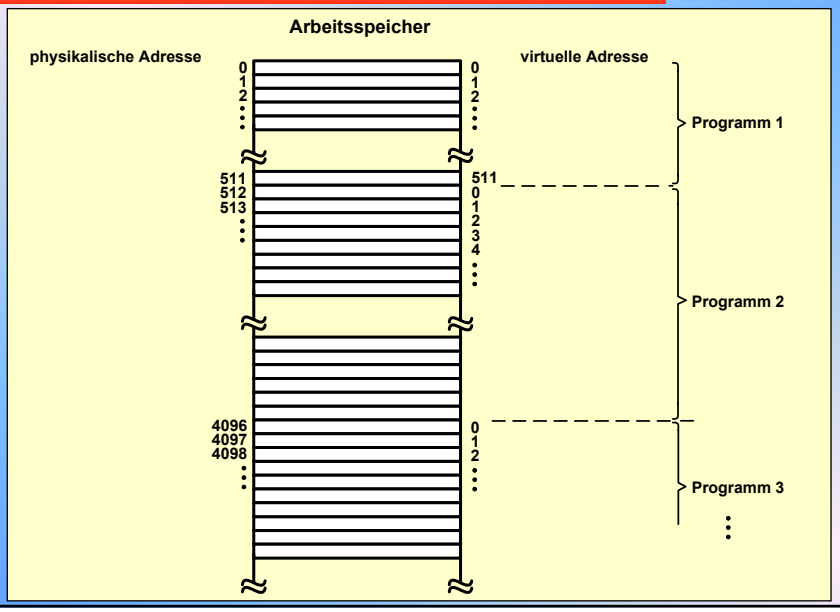
↓ Linker

Adresse relativ zum gebundenen Programm
(virtuelle Adresse)

↓ Loader

physikalische Adresse

Abbildung virtueller Adressen auf physikalische Adressen im Speicherbereich



Ein-/Ausgabesteuerung

Verschiedenartige Typen von Ein-/Ausgabegeräten

- **Unterscheidung in Geschwindigkeit**
- **Unterscheidung in Datenformaten**

Realisierung der Ein-/Ausgabesteuerung

Schnittstelle hardwareabhängig/hardwareunabhängig

- Hardwareunabhängige Ebene für die Datenverwaltung und den Datentransport
- Hardwareabhängige Ebene, die alle gerätespezifischen Eigenschaften berücksichtigt (Treiber-Programme)

Behandlung von Interrupts

- Erzeugung und Verarbeitung vektorisierter Interrupts

Interruptvektor

- Anstoß einer Interrupt-Routine bei gleichzeitiger Unterbrechung des gerade laufenden Rechenprozesses
- Priorisierung von Interrupts
- Hardwarefunktionen für die Interrupt-Behandlung (Mikrosekunden-Bereich)

Fehlerbehandlung und (Wieder-) Anlauf

Klassifizierung von Fehlern (1)

- **fehlerhafte Benutzereingaben**
  - nicht zulässige Eingaben müssen mit Fehlerhinweisen abgelehnt werden
- **fehlerhafte Anwenderprogramme**
  - Gewährleistung, dass ein fehlerhaftes Anwenderprogramm keine Auswirkungen auf andere Programme hat

Klassifizierung von Fehlern (1)

- Hardwarefehler/-ausfälle
  - Erkennung von Hardwarefehlern bzw. -ausfällen
  - Rekonfigurierung ohne die fehlerhaften Teile
  - Abschaltsequenzen bei Stromausfällen
- Deadlocks aufgrund dynamischer Konstellationen
  - sichere Vermeidung von Deadlocks ist nicht möglich

  - **Deadlockerkennung mit Deadlockbehebung durch Entzug von Betriebsmitteln**

5.3 Entwicklung eines Mini-Echtzeit-Betriebssystems

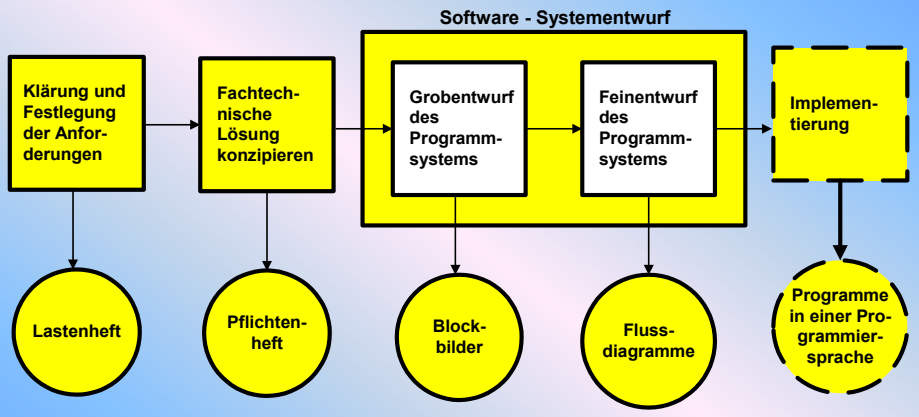
Zielsetzung

- Vorstellung des Aufbaus und der Arbeitsweise eines Echtzeit-Betriebssystems in stark vereinfachter Form
- Schrittweiser Verzicht der getroffenen Vereinfachungen

Vorgehen bei der Entwicklung

- Klärung der Aufgabenstellung, Festlegung der Anforderungen
- Fachtechnische Lösungskonzeption
- Software-Systementwurf
- Implementierung

Vorgehen bei der Entwicklung des Mini-Echtzeit-Betriebssystems



Klärung und Festlegung der Aufgabenstellung und der Anforderungen

- **Verwaltung von maximal n Rechenprozessen**
  - m Rechenprozesse zyklisch
  - k Rechenprozesse durch Interrupt-Signal

  d.h. $n = m + k$
- **ein Prozessor zur Ausführung von Operationen**
  - keine Optimierung der Abläufe durch Simultanabarbeitung von Rechen- und E/A-Operationen
- **Zeitsignal für zyklische Aktivierung durch internen Taktgeber**
  - Taktimpulse in festem Zeitabstand T (z.B. T = 20ms)
  - unterschiedliche Zykluszeiten für die zyklischen Rechenprozesse

Vereinfachungen, die später stufenweise aufgehoben werden

❶ Summe aller Rechenzeiten der Rechenprozesse kleiner als Zeitabstand T

- ❋ Sicherstellung, dass beim nächsten Uhrimpuls-Takt alle Rechenprozesse beendet sind

❷ Keine Verwaltung von Tasks, die durch Interrupt angestoßen werden

nur zyklische Tasks

❸ Keine Betriebsmittelverwaltung

- ❋ E/A-Zeiten vernachlässigbar klein

Entwurf einer Lösungskonzeption

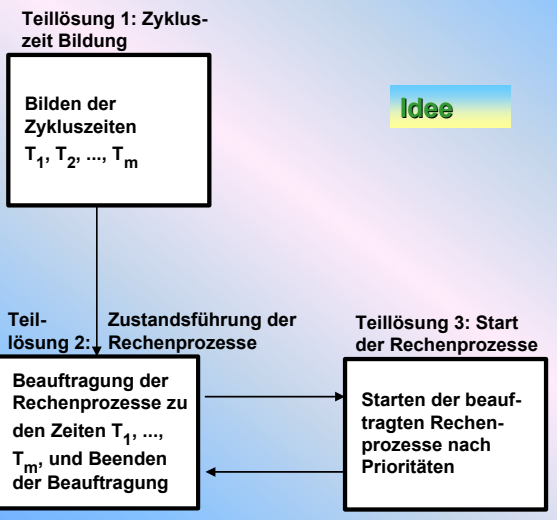
Verfahren der asynchronen Programmierung

- ❒ asynchrone Beauftragung der einzelnen Rechenprozesse
- ❒ keine feste Reihenfolge der Tasks
- ❒ Konfliktstrategie nach Prioritätsnummern

Teillösungen

- ❒ Zykluszeitbildung
  - ❋ Ableitung der unterschiedlichen Zykluszeiten der Tasks aus dem Uhrimpuls-Takt
- ❒ Zustandsführung der Rechenprozesse
  - ❋ Beauftragung der Tasks zu den jeweiligen Zykluszeiten und definierte Beendigung
- ❒ Start der Rechenprozesse
  - ❋ Start der Task, die an der Reihe ist

Lösungskonzept des Mini-Betriebssystems



Zykluszeit-Bildung

Aufgabe:

Bildung des Zusammenhangs zwischen den Zykluszeiten T_i (i = 1,2,...,m) und der Zeitdauer T

Annahme: $T_i >> T$

$\Rightarrow \quad T_i = a_i \cdot T$

a_i ganzzahlige Zykluszeitfaktoren (i=1,2,...,m)

Zeitdauervariable Z_i (i=1,2,...,m)

Eintreffen des Uhrimpuls-Takts verringert Z_i um 1

$Z_i = 0$: Zykluszeit T_i ist abgelaufen

Zurücksetzen von Z_i auf den Anfangswert a_i

Teillösung 1: Zyklus zeit-Bildung

Einführung dimensionsloser, ganzzahliger *Zykluszeit-Faktoren*

$T_1 = a_1 T$ $\quad a_1 = \frac{T_1}{T}$

$T_2 = a_2 T$ $\quad a_2 = \frac{T_2}{T}$

$T_m = a_m T$ $\quad a_m = \frac{T_m}{T}$

Definition von (dimensionslosen) Zeitdauervariablen $z_1, z_2, ..., z_m$ mit dem Anfangswerten

$z_1 = a_1$

$z_2 = a_2$

$z_m = a_m$

Uhrimpuls-takt

Bei jedem Eintreffen des Uhrimpuls-Taktes im Zeitstand T: Dekrementieren der Variablen z_i (vermindern um 1), d.h. Bildung von:

$z_1 := z_1 - 1$

$z_2 := z_2 - 1$

$z_m := z_m - 1$

Sobald eine Variable $z_i = 0$ geworden ist, ist die betreffende Zykluszeit T_i erreicht. Daher wird dieses Ergebnis an die Lösungskomponente 2 gemeldet, die den Rechenprozess i beauftragt (in den Zustand "bereit" bringt).

Zu/ von Teil-lösung 2

Rücksetzen der betreffenden Zeitdauervariablen auf den Anfangswert $z_i := a_i$, Fortsetzen mit Lösungskomponente 2.

Zu Teil-lösung 2

Zustandsführung der Rechenprozesse

Aufgabe:

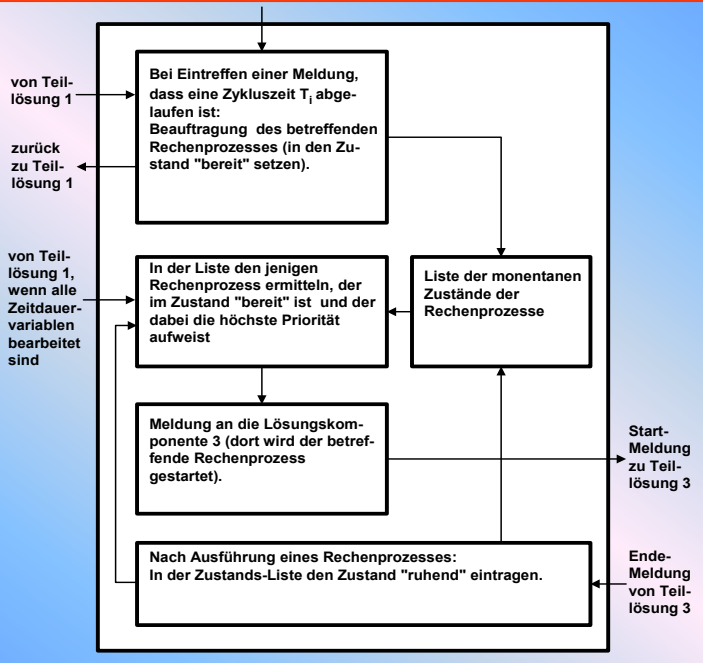
Verwaltung der Zustände der Rechenprozesse

- ruhend
- bereit
- blockiert
- laufend

Buchführung über die jeweiligen Zustände

Durchführung von Zustandsänderungen

Teil-lösung 2: Zustands-führung der Rechen-prozesse

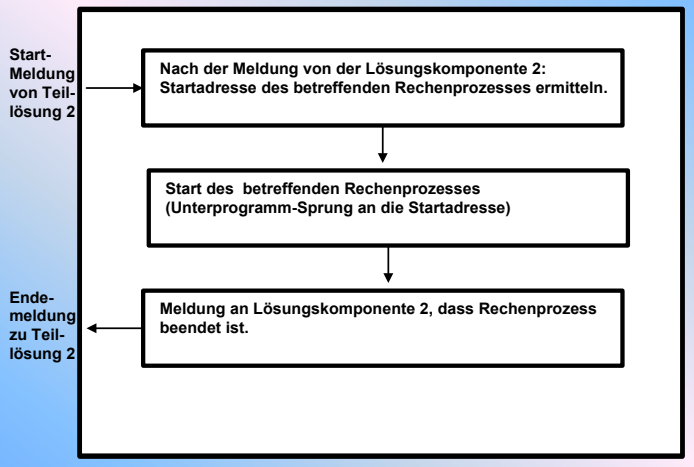


Start der Rechenprozesse

Aufgabe:

- Ermittlung der Startadresse
- Start des Rechenprozesses
- Beendigung des Rechenprozesses überwachen

Teillösung 3: Start der Rechenprozesse



5.4 Softwaresystementwurf des Mini-Echtzeit-Betriebssystems

Softwaresystementwurf unter Zugrundelegung der stark vereinfachten Aufgabenstellung

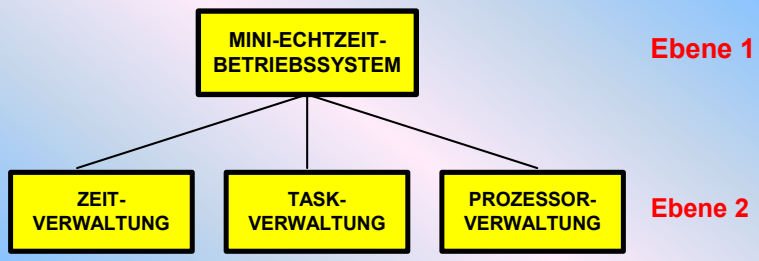
Prinzip der schrittweisen Verfeinerung

Zerlegung des Mini-Betriebssystemprogramms in Programmteile, die dann wiederum verfeinert werden

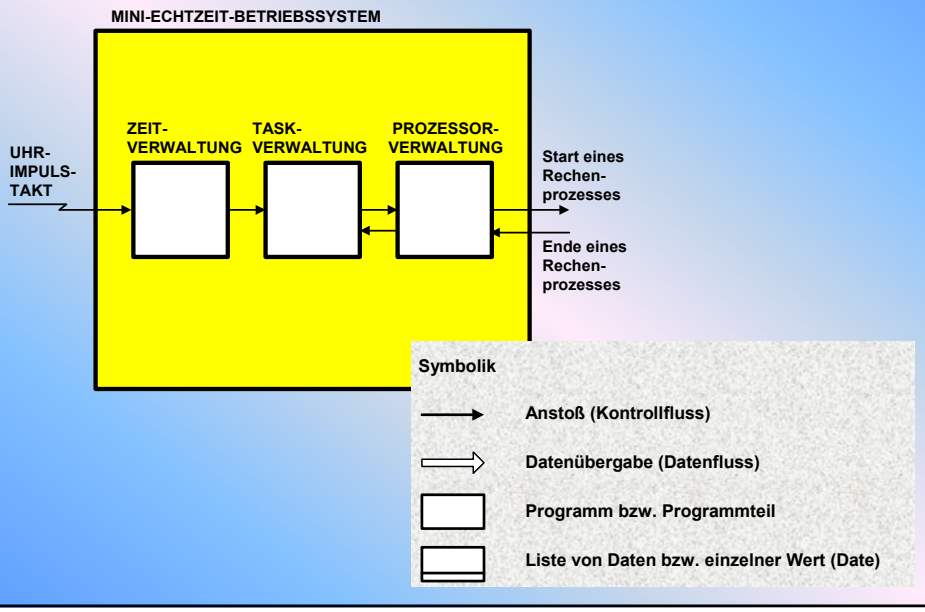
Zerlegung des Mini-Echtzeit-Betriebssystemprogramms

- Teilprogramm ZEITVERWALTUNG
  - zur Bildung der unterschiedlichen Zykluszeiten
- Teilprogramm TASKVERWALTUNG
  - zur Verwaltung der Rechenprozesse
- Teilprogramm PROZESSORVERWALTUNG
  - Zuteilung des Betriebsmittels “Prozessor”
  - Start der Rechenprozesse

Aufteilung des Mini-Echtzeit-Betriebssystems in 3 Teilprogramme



Zusammenwirken der Teilprogramme des Mini-Echtzeit-Betriebssystems



Benötigte Listen für die ZEITVERWALTUNG

- Zeitdauervariable Z_i zur Zykluszeit-Bildung

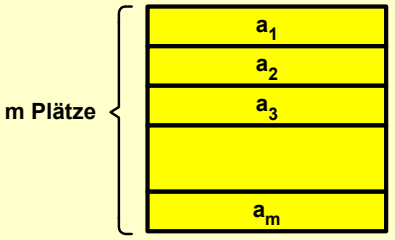
  Liste ZEITZÄHLER

- Zykluszeit T_i für jeden Rechenprozess

  Liste ZYKLUS

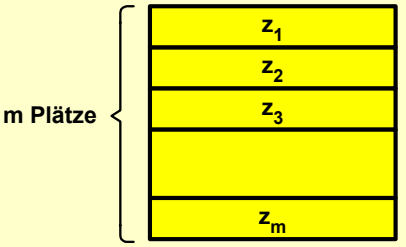
Aufbau der Listen für die Zeitverwaltung

Bereitstellen der Zykluszeit-Faktoren a_i in einer Liste ZYKLUS:



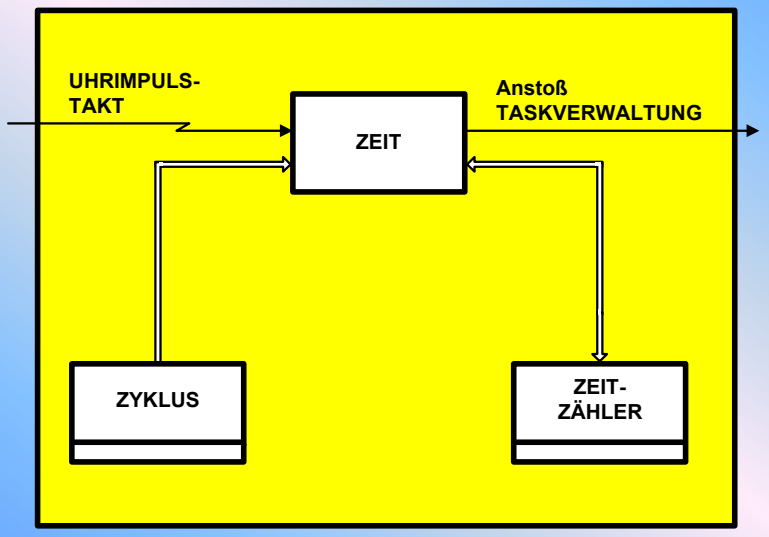
= T_i/T
Liste von Konstanten

Ablage der Zeitdauervariablen z_i in einer Liste ZEITZÄHLER:



Liste von Variablen

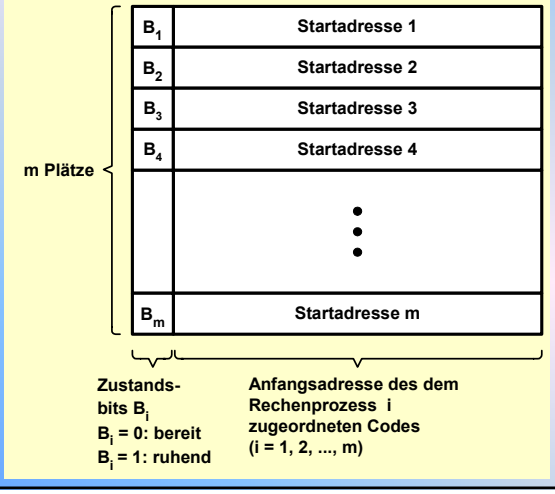
Blockdiagramm des Teilprogramms ZEITVERWALTUNG



Benötigte Liste für TASKVERWALTUNG und PROZESSORVERWALTUNG

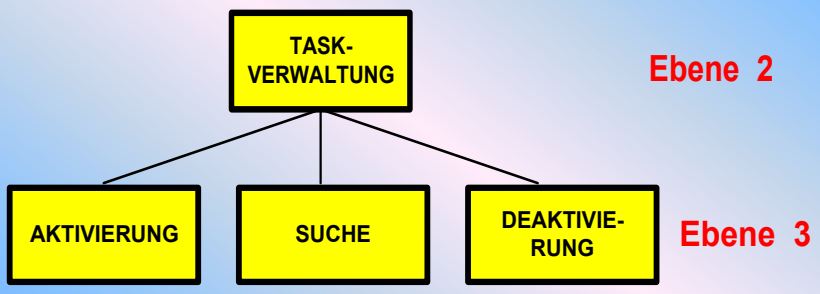
⇨ Zustände und Anfangsadressen der Rechenprozesse Liste VERWALTUNGSBLOCK

Listenstruktur zur Verwaltung der Rechenprozesse VERWALTUNGSBLOCK

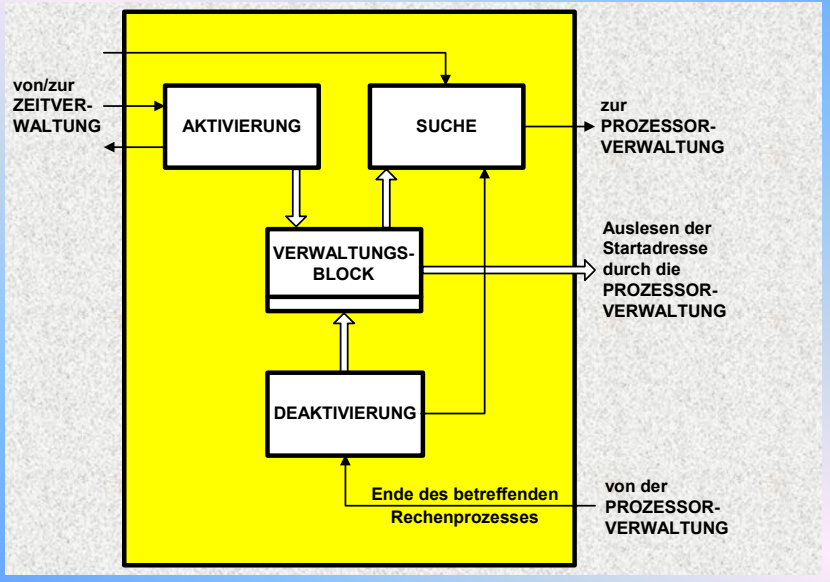


zwei-dimensionales Feld

Zerlegung des Programmteils TASKVERWALTUNG



Blockdiagramm der TASKVERWALTUNG

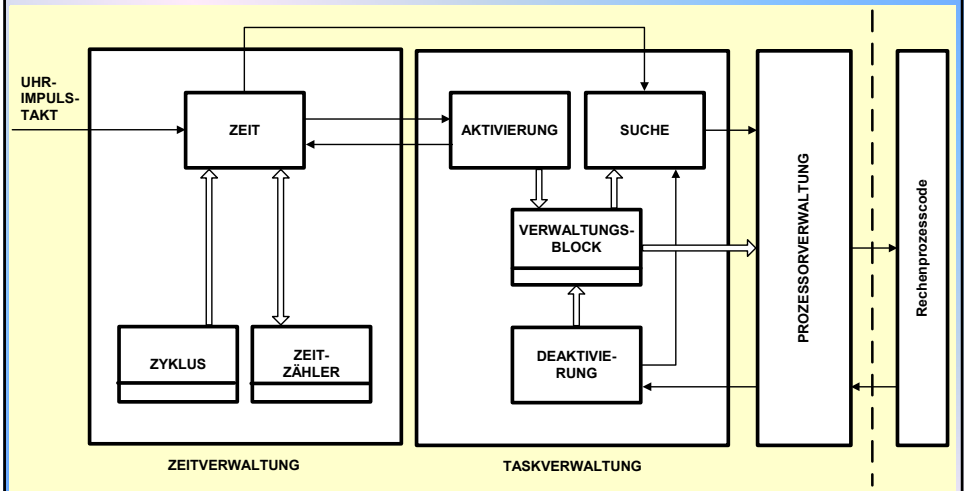


AKTIVIERUNG:	Verändern des Zustandsbits in “bereit”
DEAKTIVIERUNG:	Verändern des Zustandsbits in “ruhend”
SUCHE:	Überprüfung, ob sich eine Task im Zustand “bereit” befindet

Ordnung der Liste VERWALTUNGSBLOCK ermöglicht einfache Priorisierung

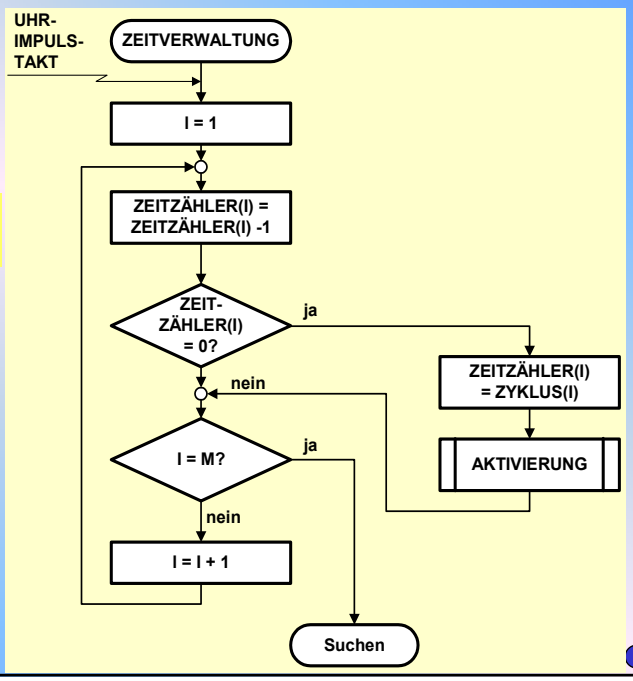
Untergliederung PROZESSORVERWALTUNG ist nicht notwendig

Übersichtsblockdiagramm des Mini-Echtzeit-Betriebssystems

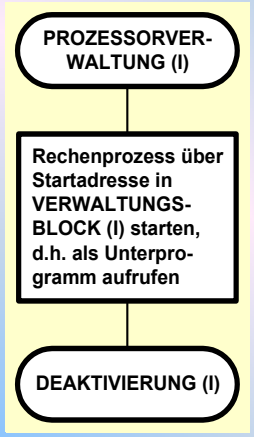


Feinentwicklung in Form von Flussdiagrammen

Flussdiagramm der ZEITVERWALTUNG

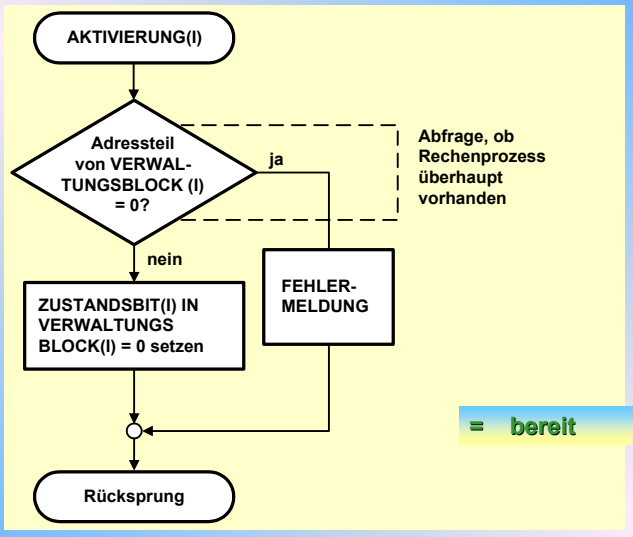


Flussdiagramm der PROZESSORVERWALTUNG



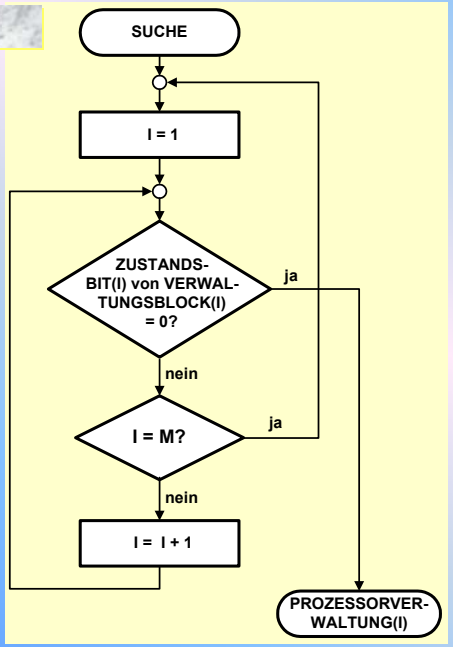
Flussdiagramme der Programme TASKVERWALTUNG

AKTIVIERUNG

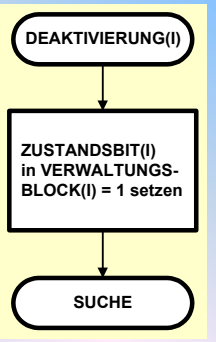


= bereit

Suche



DEAKTIVIERUNG



= ruhend

Erste Erweiterung des Systementwurfs

Zulassen längerer Rechenzeiten für die Rechenprozesse

⇨ Beim Eintreffen eines Uhrimpuls-Taktes muß u.U. ein noch laufender Rechenprozess mit längerer Ausführungsdauer und niedrigerer Priorität unterbrochen werden, um einen Rechenprozess höherer Priorität zu starten

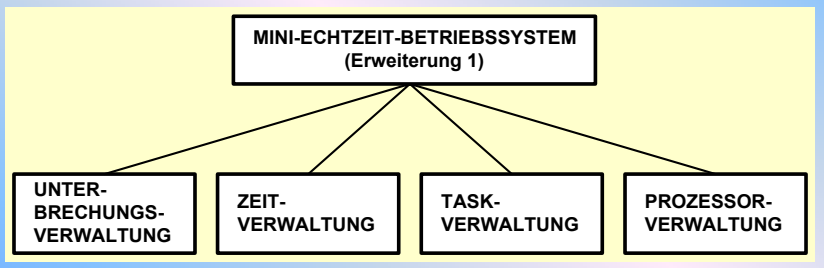
Teilprogramm UNTERBRECHUNGSVERWALTUNG (Verwaltungsprogramm)

Aufgabe:

⇨ Rettung der Register des Prozessors eines gerade laufenden Rechenprozesses

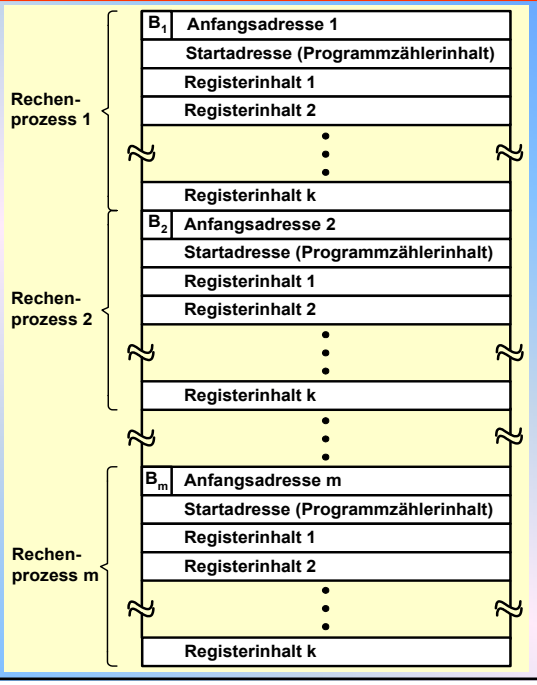
- Programmzähler
- Akkumulator
- Zustandsregister
- Arbeitsregister

Erweitertes Hierarchiediagramm nach dem Zulassen längerer Rechenzeiten der Rechenprozesse



Erweiterung der Liste VERWALTUNGS-BLOCK

- Startadresse nach einer Unterbrechung
- Register-Speicherplätze



Erweiterung des Teilprogramms PROZESSORVERWALTUNG

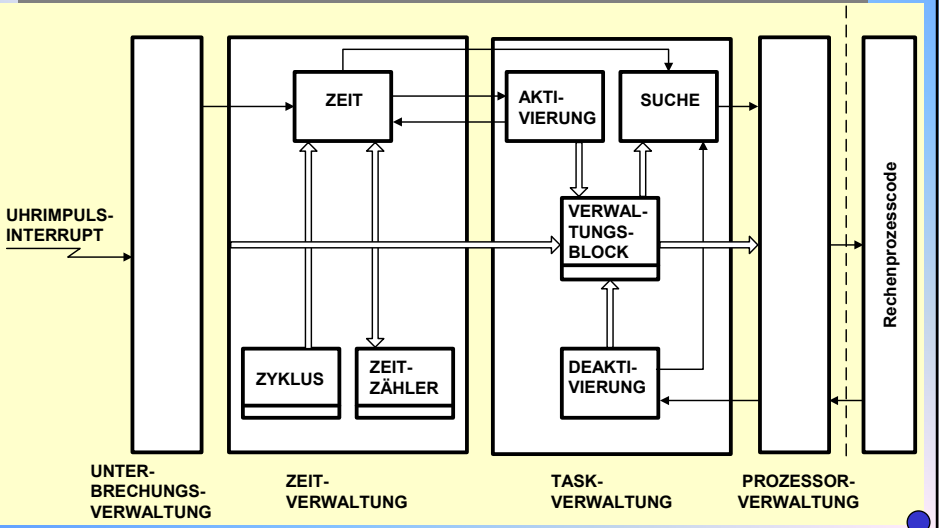
⇨ vor dem Start eines ablaufbereiten Rechenprozesses laden der Register mit den Inhalten der Liste VERWALTUNGSBLOCK

Erweiterung des Teilprogramms DEAKTIVIERUNG

⇨ nach Beendigung eines Rechenprozesses laden seiner Anfangsadresse in die Zelle STARTADRESSE und Initialisieren der Registerinhalte im VERWALTUNGSBLOCK

Übersichtsdiagramm des Mini-Echtzeit-Betriebssystems

Erste Erweiterung: Zulassen längerer Rechenzeiten für die Rechenprozesse



Zweite Erweiterung des Software-Systementwurfs

Vorsehen der Möglichkeit von Alarm-Interrupts

⇨ bis zu k Rechenprozesse, deren Aktivierung von zeitlich nicht vorhersehbaren Alarm-Interrupts ausgelöst wird

Erweiterung der UNTERBRECHUNGSVERWALTUNG

⇨ Registerrettung

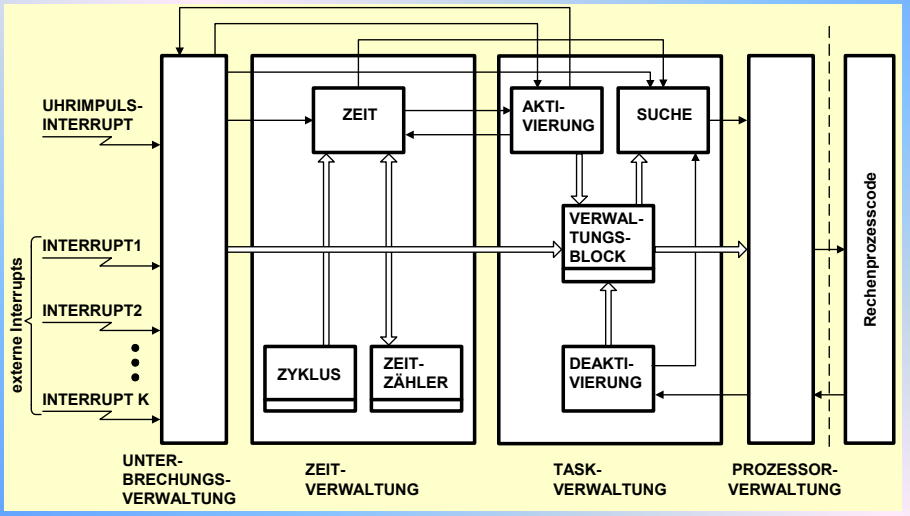
⇨ bei Uhrimpuls-Interrupts Anstoß der ZEITVERWALTUNG

⇨ bei Alarm-Interrupts Aufruf der AKTIVIERUNG, um dem Alarm-Interrupt zugeordnetes Antwortprogramm in den Zustand “bereit” zu setzen

⇨ Anstoß der SUCHE

Übersichtsdiagramm für das Mini-Betriebssystem

Zweite Erweiterung: Vorsehen der Möglichkeit von Alarm-Interrupts



Dritte Erweiterung des Software-Systementwurfs

Betriebsmittelverwaltung für Ein-/Ausgabegeräte

E/A-Operationen sind langsamer

⇨ Analog-Digital-Umsetzer ca. 20 ms

Einführung eines Verwaltungsprogramms E/A-VERWALTUNG

Aufgabe:

Organisation von langsamen Ein-/Ausgabeoperationen

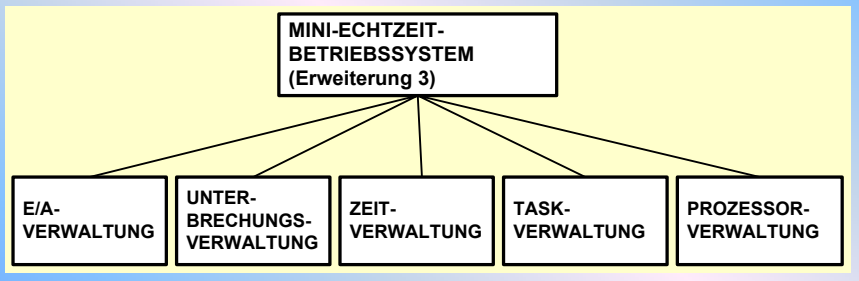
⇨ Rechenprozess wird angehalten

⇨ Prozessor kann andere Rechenprozesse bearbeiten

⇨ Beendigung der E/A-Operationen ermöglicht Fortsetzung des zugehörigen Rechenprozesses

Hierarchiediagramm des Mini-Betriebssystems

Dritte Erweiterung

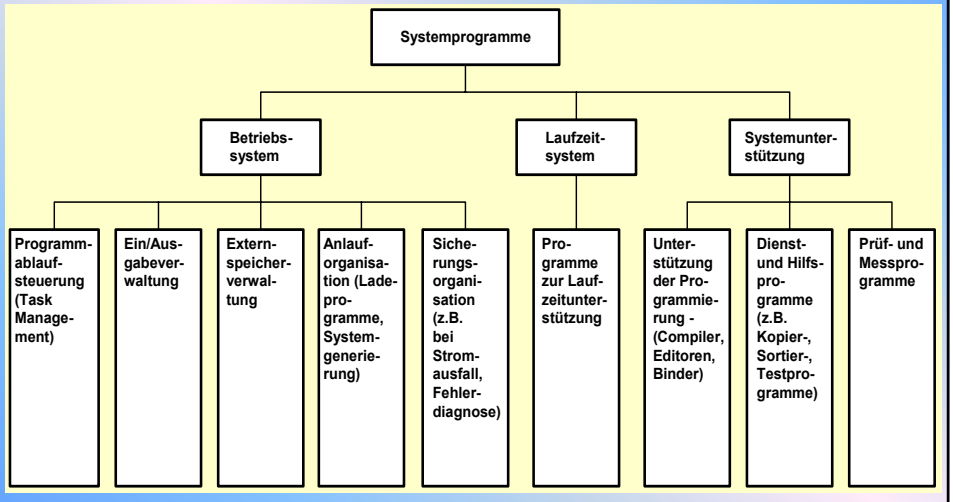


Übergang zum realen Echtzeit-Betriebssystem

Aufhebung der Vereinfachungen

- Die Betriebssystemprogramme selbst sind nicht unterbrechbar
- Die Mehrfachbeauftragung eines Rechenprozesses, d.h. erneute Beauftragung vor der Beendigung, ist ausgeschlossen
- Eine gegenseitige Beauftragung von Rechenprozessen ist nicht möglich
- Eine Synchronisierung der Rechenprozesse, z.B. mittels Semaphoroperationen, ist nicht möglich
- Keine Datenkommunikation zwischen den Rechenprozessen, d.h. kein Austausch von Daten, keine gemeinsame Benutzung von Daten
- Keine dynamische Änderung der Prioritäten der Rechenprozesse während der Programmdurchführung
- Rechenprozesse befinden sich im Arbeitsspeicher, Hintergrundspeicher sind nicht vorhanden

5.5 Gliederung der Systemprogramme in anwendungsbezogene Programmbausteine



5.6 Beispiele für Echtzeit-Betriebssysteme

Marktübersicht

Kriterien bei der Auswahl von Echtzeit-Betriebssystemen

- Entwicklungs- und Zielumgebung
- Modularität und Kernelgrösse
- Leistungsdaten
  - Anzahl von Tasks
  - Prioritätsstufen
  - Taskwechselzeiten
  - Interruptlatenzzeit

- Anpassung an spezielle Zielumgebungen
- Allgemeine Eigenschaften
  - Schedulingverfahren
  - Intertaskkommunikation
  - Netzwerkkommunikation
  - Gestaltung Benutzungsoberfläche

Auswahl kommerzieller Echtzeit-Betriebssysteme

Produkt	ERCOS	Lynx-OS	OS/9	OSE Delta	pSOS	PXROS	QNX	VRTX32	VxWorks	Windows CE
Hersteller	ETAS GmbH	Lynx Real-Time System inc.	Microware	ENEA DATA AB	ARS Integrated Systems	HighTec EDV-Systeme	QNX Software Systems LTD	Microtec Research	WindRiver	Microsoft
Kategorie	Embedded		EZBS, EZK, Embedded	EZK, Embedded	EZBS, EZK, Embedded	EZK, Embedded	EZBS, EZK, Embedded	EZBS, EZK, Embedded	EZBS, EZK, Embedded	EZBS Embedded
Ziel-system	8016x, PowerPC	680x0, 80x86, PowerPC, 88000, i860, MIPS, SPARC, RS6000	680x0, 80x86, PowerPC, CPU32	680x0, PowerPC, CPU32, AMD29k	680x0, 80x86, 8016x, PowerPC, CPU32, i960, Hitachi SH, MIPS	80x86, 8016x, PowerPC	i386,i486, Pentium, 80286(16 bit)	680x0, 80x86, SPARC, CPU32, AMD29k, i960	680x0, 80x86, PowerPC, CPU32, i960, MIPS, SPARC, AMD29k, Hitachi SH	Pentium 80x86, i486 PowerPC MIPS Hitachi S4, ARM
Host-system	UNIX, Win95, NT	UNIX	UNIX, Windows	UNIX, Windows, NT	UNIX, SUN, Windows, NT, OS/2	UNIX, SUN, Windows, NT, OS/2	QNX	UNIX, SUN, Windows	UNIX, Win95, NT	Windows CE Win 95 NT
Sprache	ANSI-C, OLT Specification Language	ANSI-C, C++, Pascal, Ada, Modula, Fortran	ANSI-C, C++	C, C++	ASM, ANSI-C, C++, Pascal, Ada	ANSI-C, C++	Watcom C, C++, Inline ASM	ASM, ANSI-C, C++	ANSI-C, C++, Java, Ada	Visual C++ Visual Basic Visual J++
Datei-system	kein	UNIX, FAT, NFS, Real-Time Filesystem	FAT	UNIX, FAT	UNIX, FAT, NFS, Real-Time Filesystem	UNIX, FAT	UNIX, FAT, ISO9660	UNIX, FAT	UNIX, FAT	FAT
GUI-System	kein	X-Win	X-Win	kein	X-Win, MetaWin for RT	X-Win	X-Win (Motif), QNX-Win, Photon		X-Win	Windows CE

Fortsetzung

Produkt	ERCOS	Lynx-OS	OS/9	OSE Delta	pSOS	PXROS	QNX	VRTX32	VxWorks	Windows CE
Netzwerk		TCP/IP, NFS	TCP/IP, OS/9-net, NeWLink	TCP/IP, PPP, SNMP	TCP/IP, Netware, OSI 1-7, SNMP CMIP, X.25	TCP/IP, NFS	TCP/IP, NFS, SNMP, Streams	TCP/IP, Netware	TCP/IP, NFS, SNMP, Streams	TCP/IP, PPP bzw. SLIP
Feldbus	CAN		CAN, PROFI-BUS, Interbus-S		CAN	CAN, PROFI-BUS		CAN, PROFI-BUS, LON		
Sonstiges	ROM-fähig	ROM-fähig, Multipro-zessor, self-hosted	ROM-fähig, Multipro-zessor	ROM-fähig, Multipro-zessor	ROM-fähig, Multipro-zessor, fehlertoler ant	ROM-fähig, Multipro-zessor	ROM-fähig, Multipro-zessor, POSIX 1003 kompatibel	ROM-fähig	ROM-fähig, Multipro-zessor, POSIX 1003 kompatibel	ROM-fähig
Sche-duling	preemptiv, kooperativ, prioritäts-gesteuert	preemptiv, prioritäts-gesteuert, Round-Robin	preemptiv, kooperativ, prioritäts-gesteuert, Round-Robin	preemptiv,	preemptiv, prioritäts-gesteuert, Round-Robin	preemptiv, prioritäts-gesteuert	preemptiv, prioritäts-gesteuert, Round-Robin	preemptiv, prioritäts-gesteuert, Round-Robin	preemptiv, prioritäts-gesteuert, Round-Robin	preemptiv, prioritäts-gesteuert
Task-wech-selzeit	< 54 µs 8016x (20 MHz)						4,7µs Pentium 166, 11,1us 486DX4 (100MHz), 74,2us 386 (33MHz)	17 µs		≥ 100 µs

Echtzeit-Betriebssystem QNX

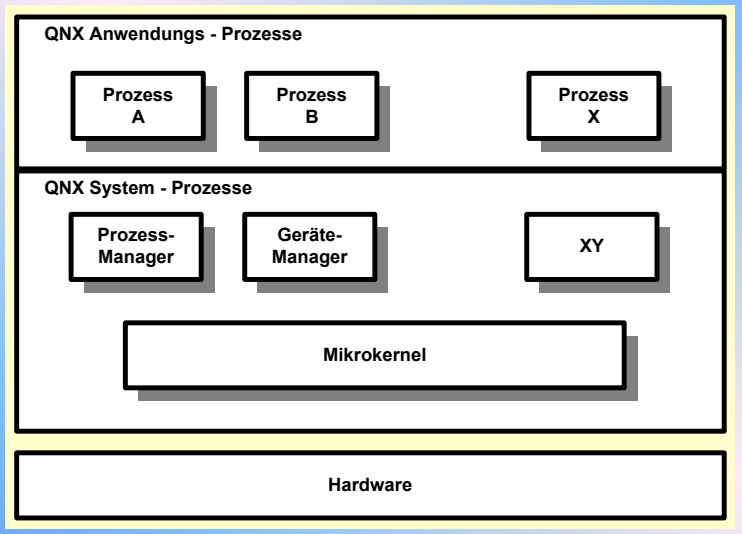
Eigenschaften

- Echtzeitbetriebssystem bzw. Echtzeitkernel
- Mikrokernel <= 8 kByte
- Systemprozesse wie z.B. Prozess-Manager

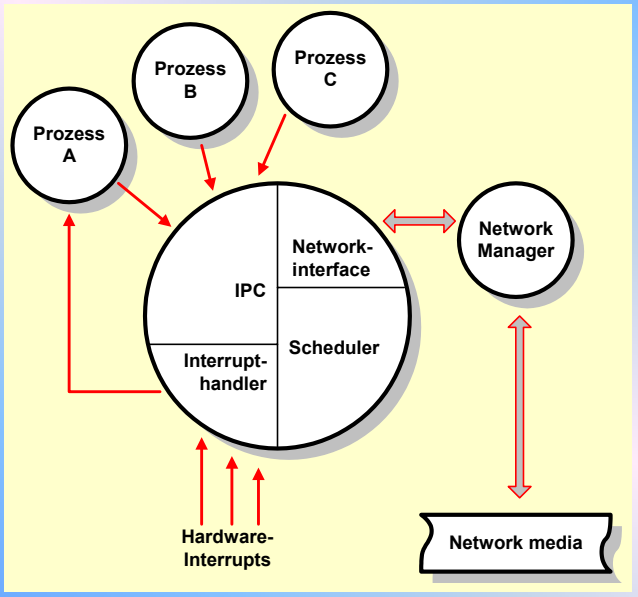
Typische QNX-Konfigurationen

- Mikrokernel
- Prozess-Manager
- Geräte-Manager
- Dateisystem-Manager
- Netzwerk-Manager

Anwendungs- und Systemprozesse unter dem Echtzeit-Betriebssystem QNX



Mikrokernel von QNX



Prozesszustände und Zustandsübergänge von QNX-Prozessen