OSEK / OSEKtime
Ausgewählte Kapitel eingebetteter Systeme

Wilhelm Haas

Wilhelm.Haas@informatik.stud.uni-erlangen.de

Friedrich-Alexander-Universität Erlangen-Nürnberg
Institut für Informatik
Lehrstuhl 4

15. Juli 2006

Überblick

OSEK-Gremium

- OSEK $\hat{=}$ *Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug*
- 1993 gegründet
- industrielles Standardisierungsgremium
- Gründungsmitglieder:
  - BMW AG
  - Siemens AG
  - Daimler-Benz AG
  - ...
- 1994 Zusammenschluss mit VDX[1]-Initiative zu OSEK/VDX

[1] Vehicle Distributed Executive

OSEK-Standards

Geschaffene Standards:

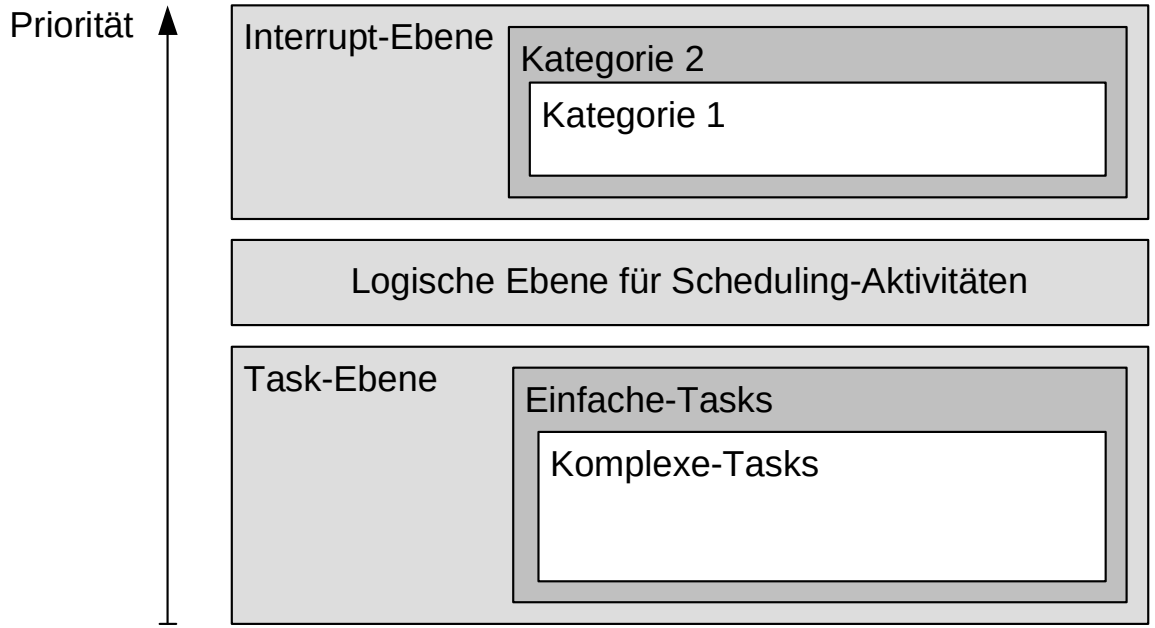
- **OSEK-OS:** Event Triggered Realtime Operating System
- **OSEKtime-OS:** Time Driven Realtime Operating System
- **OSEK-OIL:** OSEK **I**mplementation **L**anguage
- **OSEK-ORTI:** OSEK **R**un**T**ime **I**nterface
- **OSEK-COM:** **COM**munication
- **OSEK-NM:** OSEK **N**etwork **M**anagement

Überblick

Architektur

OSEK-OS:

- statisches, ereignisgesteuertes Betriebssystem
  - Betriebsmittel und Tasks müssen noch während der Entwicklung konfiguriert werden
- 4 Konformitätsklassen legen fest:
  - welche Task-Typen verwendet werden
  - Anzahl der Aktivierungen eines Tasks
  - Anzahl Tasks pro Prioritätsebene



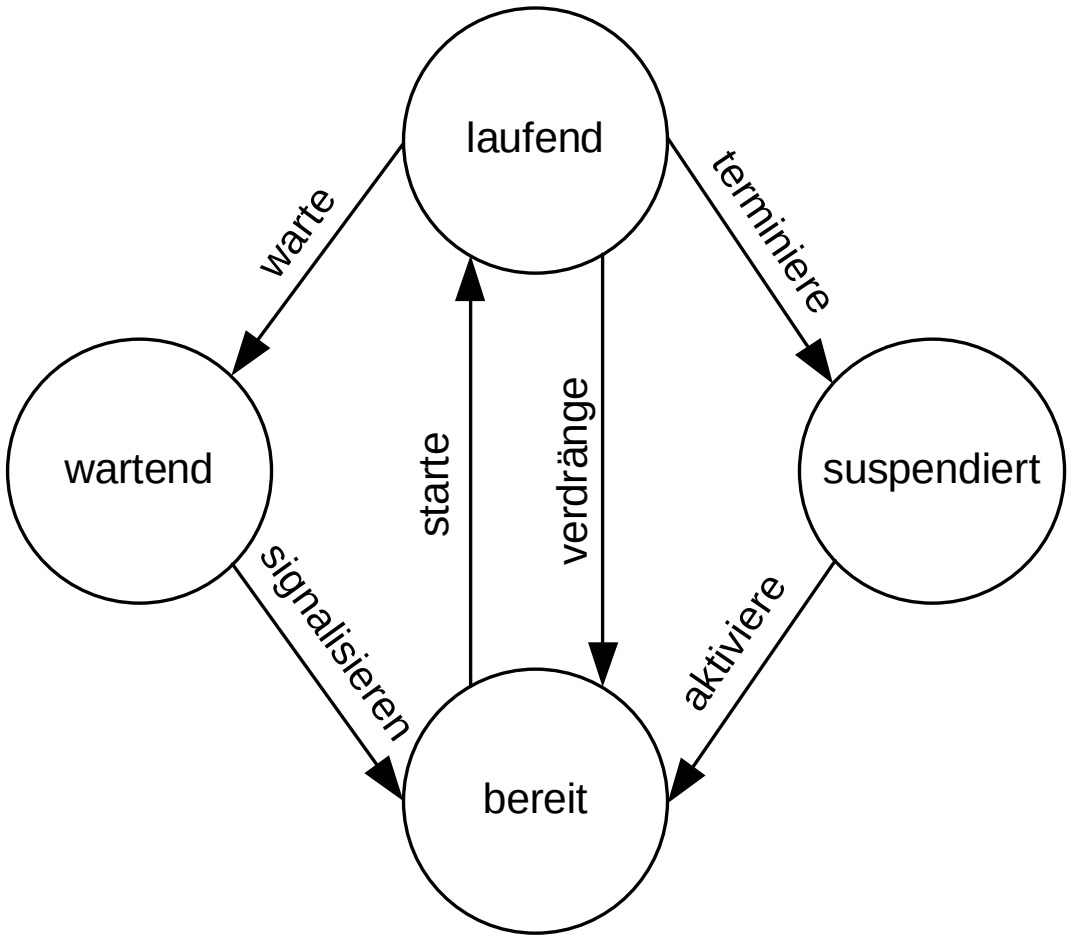
- 3 Prozesse:
  - Prozess für die Interrupt-Behandlung (höchste Priorität)
  - Prozess für den Scheduler
  - Prozess für die Abarbeitung der Tasks (niedrigste Priorität)

Task-Management

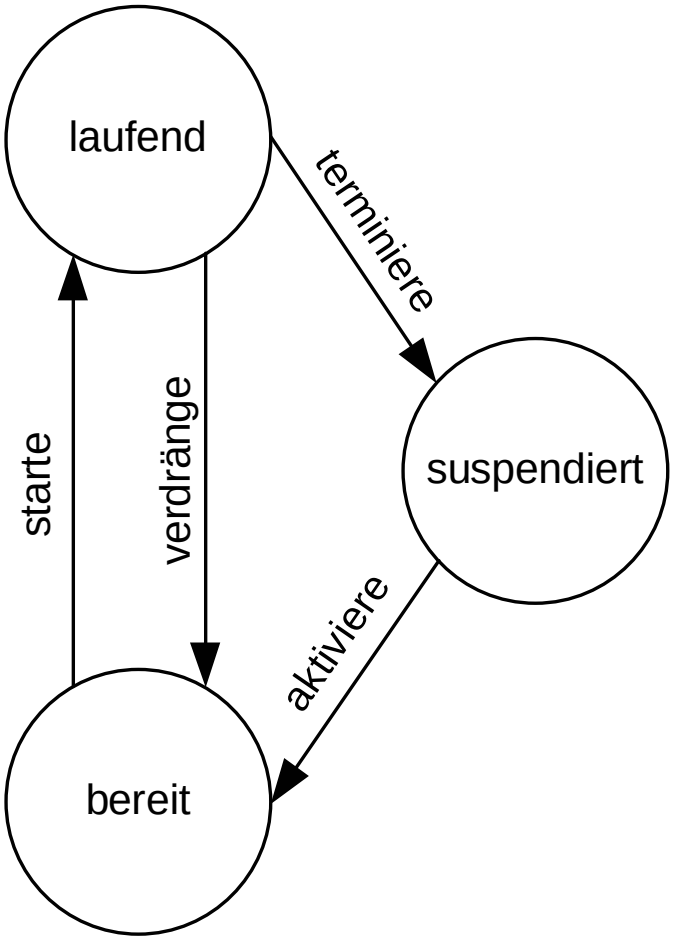
OSEK unterscheidet zwischen zwei Task-Typen:

- **Komplexe Tasks**
  - erwarten die Zuteilung von Betriebsmitteln
  - dürfen auf Ereignise warten
  - können blockieren
  - $\Rightarrow$ mehrere Synchronisationspunkte
- **Einfache Tasks**
  - laufen durch ohne zu blockieren
  - $\Rightarrow$ zwei Synchronisationspunkte

Komplexe Tasks



Einfache Tasks



Scheduling:

- Task werden nach ihren Prioritäten eingeplant
- Scheduling-Arten:
  - verdrängendes Scheduling
  - nichtverdrängendes Scheduling
  - Mischbetrieb (Mischung aus verdrängendem und nichtverdrängendem Scheduling)

Interrupt-Behandlung

Es existieren zwei Interrupt-Behandlungs (ISR) Kategorien :

- ISR der Kategorie 1
  - verwendet keine Systemfunktionen
  - hat keinen Einfluss auf den unterbrochenen Task
- ISR der Kategorie 2
  - sind für die Behandlungsroutinen des Benutzers vorgesehen
  - eingeschränkte nutzung von Systemfunktionen

- Hardware kümmert sich um die Prioritätenvergabe
- Hardware ist auch für die Einplanung zuständig
- Aktivierung und Deaktivierung der Interrupts möglich
- Fehler führen zu einem undefinierten Verhalten

Ereignismechanismus

- Ereignise sind Objekte
- dürfen nur von den komplexen Tasks verwendet werden
- sind Träger von binären Informationen
- nur vom *Besitzer* zurücksetzbar
- *BÖSE (erschwerte Antwortzeitanalyse)*

Betriebsmittelverwaltung

- Koordinierung des Zugriffs konkurierender Tasks verschiedener Prioritäten auf gemeinsame Betriebsmittel
- auch auf ISR erweiterbar
- stellt sicher, dass keine zwei Tasks zur selben Zeit die selbe Ressource besitzen

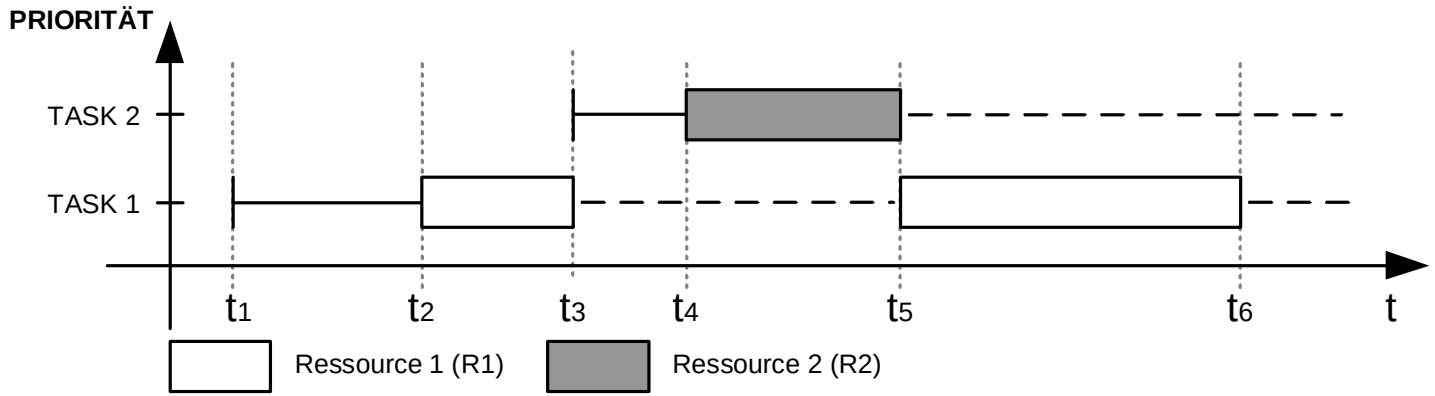
Wann ist dieser Mechanismus nützlich?

- bei verdrängbaren Tasks
- bei gemeinsamer Betriebsmittelnutzung zwischen Tasks und ISRs
- bei gemeinsamer Betriebsmittelnutzung zwischen ISRs

Probleme:

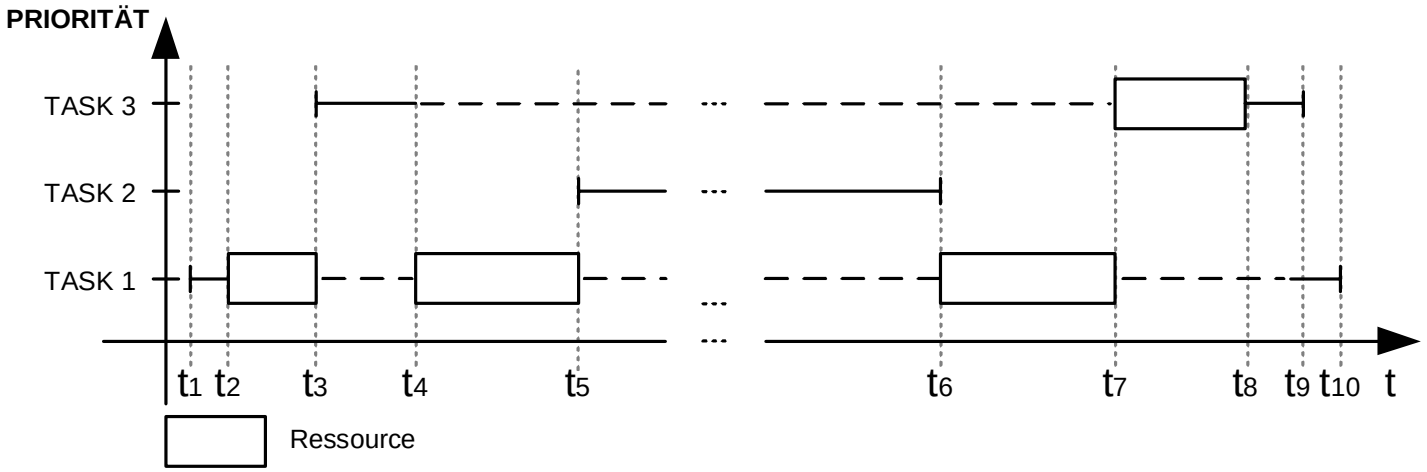
- blockierende Synchronisation
  - Deadlock
  - (unkontrollierte) Prioritätsumkehr

Beispiel für einen Deadlock



Zeit (t)	TASK 1	TASK 2
1	wird aktiviert und gestartet	
2	belegt Ressource 1 (R1)	
3	wird verdrängt	wird aktiviert und gestartet
4		belegt Ressource 2 (R2)
5	wird gestartet	möchte R1, wird blockiert
6	möchte R2, wird blockiert	

Beispiel für eine (unkontrollierte) Prioritäsumkehr



Zeit (t)	TASK 1	TASK 2	TASK 3
1	wird aktiviert und gestartet		
2	belegt Ressource 1 (R1)		
3	wird verdrängt		wird aktiviert und gestartet
4	wird gestartet		möchte R1, wird blockiert
5		wird aktiviert und gestartet	
6	wird gestartet	terminiert	
7	gibt R1 frei, wird verdrängt		wird gestartet und belegt R1
8			gibt R1 frei
9	wird gestartet		terminiert
10	terminiert		

Lösung: OSEK Priority-Ceiling-Protokoll

Vier Regeln sind einzuhalten:

1. $\mathit{Prioritätsobergrenze}(r) = \mathit{max}(T.\mathit{Priorität} : T.\mathit{belegt}(r))$
   $\forall T \in \mathit{Tasks}, \forall r \in \mathit{Betriebsmittel}$
2. Ein Task darf ein Betriebsmittel nur belegen wenn seine Priorität $\leq$ der Prioritätsobergrenze des Betriebsmittels ist.
3. Die Priorität des Tasks wird auf die Prioritätsobergrenze angehoben.
4. Freigabe der Betriebsmittel in der LIFO-Reihenfolge. Dabei wird die Priorität des Tasks herabgesetzt.

Lösung: OSEK Priority-Ceiling-Protokoll

Vier Regeln sind einzuhalten:

1. $\mathit{Prioritätsobergrenze}(r) = \mathit{max}(T.\mathit{Priorität} : T.\mathit{belegt}(r))$
   $\forall T \in \mathit{Tasks}, \forall r \in \mathit{Betriebsmittel}$
2. Ein Task darf ein Betriebsmittel nur belegen wenn seine Priorität $\leq$ der Prioritätsobergrenze des Betriebsmittels ist.
3. Die Priorität des Tasks wird auf die Prioritätsobergrenze angehoben.
4. Freigabe der Betriebsmittel in der LIFO-Reihenfolge. Dabei wird die Priorität des Tasks herabgesetzt.

Lösung: OSEK Priority-Ceiling-Protokoll

Vier Regeln sind einzuhalten:

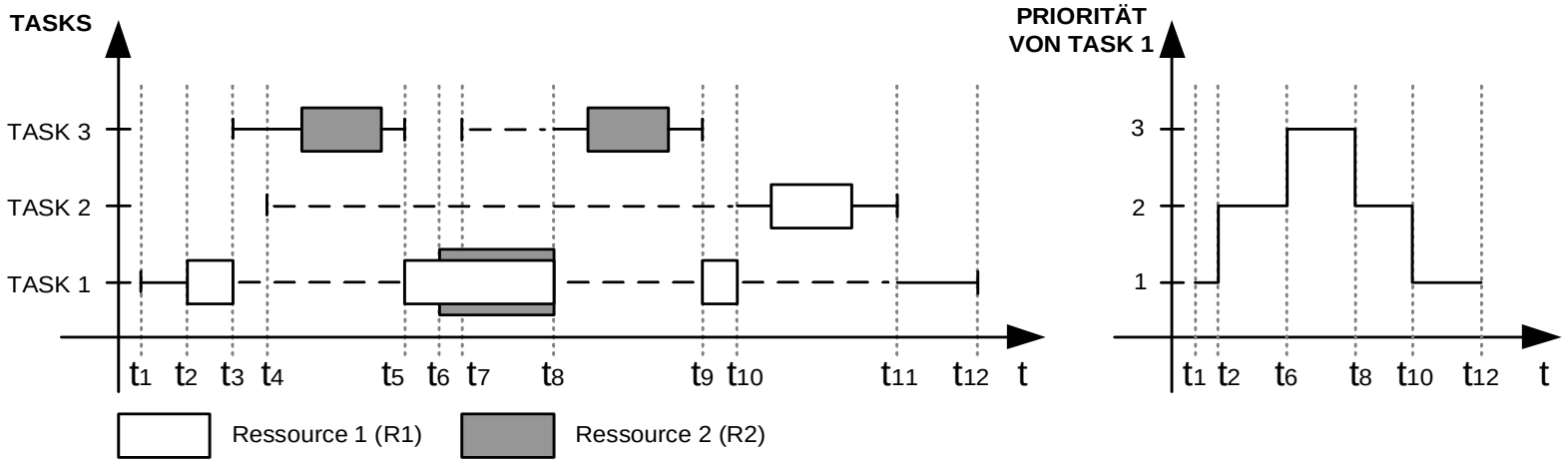
1. $\mathit{Prioritätsobergrenze}(r) = \mathit{max}(T.\mathit{Priorität} : T.\mathit{belegt}(r))$
   $\forall T \in \mathit{Tasks}, \forall r \in \mathit{Betriebsmittel}$
2. Ein Task darf ein Betriebsmittel nur belegen wenn seine Priorität $\leq$ der Prioritätsobergrenze des Betriebsmittels ist.
3. Die Priorität des Tasks wird auf die Prioritätsobergrenze angehoben.
4. Freigabe der Betriebsmittel in der LIFO-Reihenfolge. Dabei wird die Priorität des Tasks herabgesetzt.

Lösung: OSEK Priority-Ceiling-Protokoll

Vier Regeln sind einzuhalten:

1. $\mathit{Prioritätsobergrenze}(r) = \mathit{max}(T.\mathit{Priorität} : T.\mathit{belegt}(r))$
   $\forall T \in \mathit{Tasks}, \forall r \in \mathit{Betriebsmittel}$
2. Ein Task darf ein Betriebsmittel nur belegen wenn seine Priorität $\leq$ der Prioritätsobergrenze des Betriebsmittels ist.
3. Die Priorität des Tasks wird auf die Prioritätsobergrenze angehoben.
4. Freigabe der Betriebsmittel in der LIFO-Reihenfolge. Dabei wird die Priorität des Tasks herabgesetzt.

Beispiel für Priority-Ceiling-Protokoll



Zeit (t)	TASK 1	TASK 2	TASK 3
1	wird aktiviert und gestartet		
2	belegt R1, hat Priorität 2		
3	wird verdrängt		wird aktiviert und gestartet
4		wird aktiviert	
5	startet		wird terminiert
6	belegt R2, bekommt Priorität 3		
7			wird aktiviert
8	gibt R2 frei, hat Priorität 2		startet
9	startet		terminiert
10	gibt R1 frei, hat Priorität 1	wird gestartet	
11		terminiert	
12	terminiert		

Fehlerbehandlung

- verschiedene Hook-Routinen stehen zur Verfügung
- werden in abhängigkeit vom Systemzustand ausgelöst
- Hook-Routinen haben höhere Prioritäten als alle Tasks des Systems
- werden nicht von ISR der Kategorie 2 unterbrochen
- eingeschränkte Systemfunktionnutzung

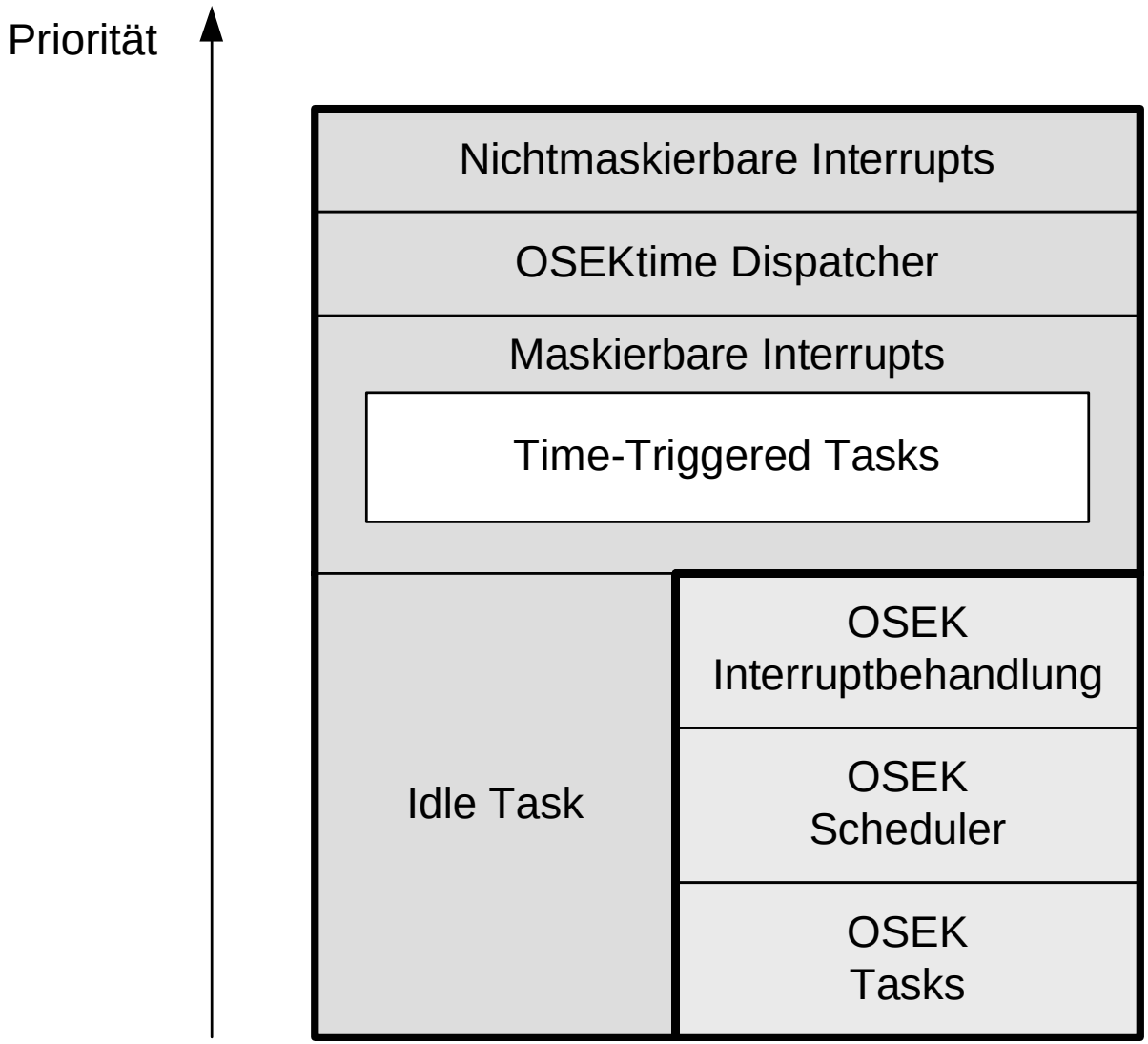
Einsatzarten:

- beim Systemstart (*StartupHook*)
- vor dem Herunterfahren des Systems (*ShutdownHook*)
- zum Debuggen und für Zeitmessungen (*PreTaskHook*, *PostTaskHook*)
- im Fehlerfall (*ErrorHook*)

Überblick

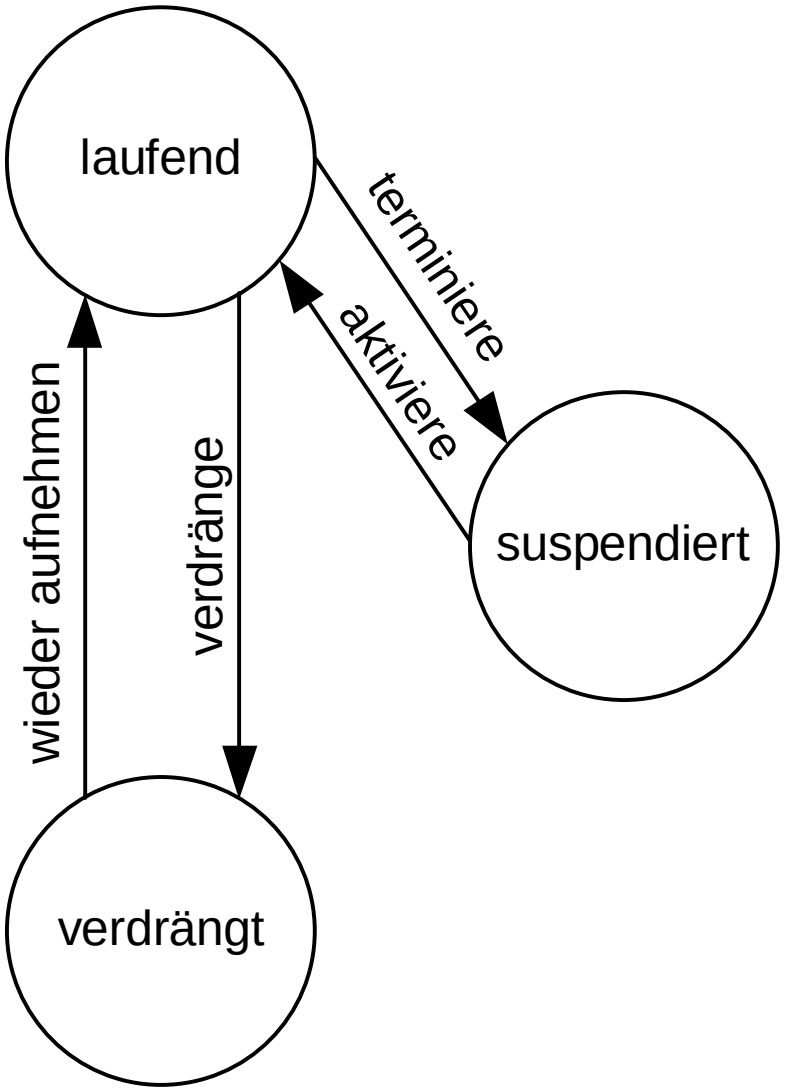
Architektur

- statisches, zeitgesteuertes Echtzeitsystem
- alle Parameter bekannt
  - Aktivierungspunkte
  - Deadlines
  - Ausführungszeiten
- Abarbeitung einer vorher berechneten Ablauftabelle


Priorität
Nichtmaskierbare Interrupts
OSEKtime Dispatcher
Maskierbare Interrupts
Time-Triggered Tasks
Idle Task
OSEK Interruptbehandlung
OSEK Scheduler
OSEK Tasks

Task-Management

- sequenzielle Ausführung nach einem statischen Ablaufplan
- laufen durch ohne zu blockieren
- verdrängbares Scheduling
- Terminierung nur durch sich selbst

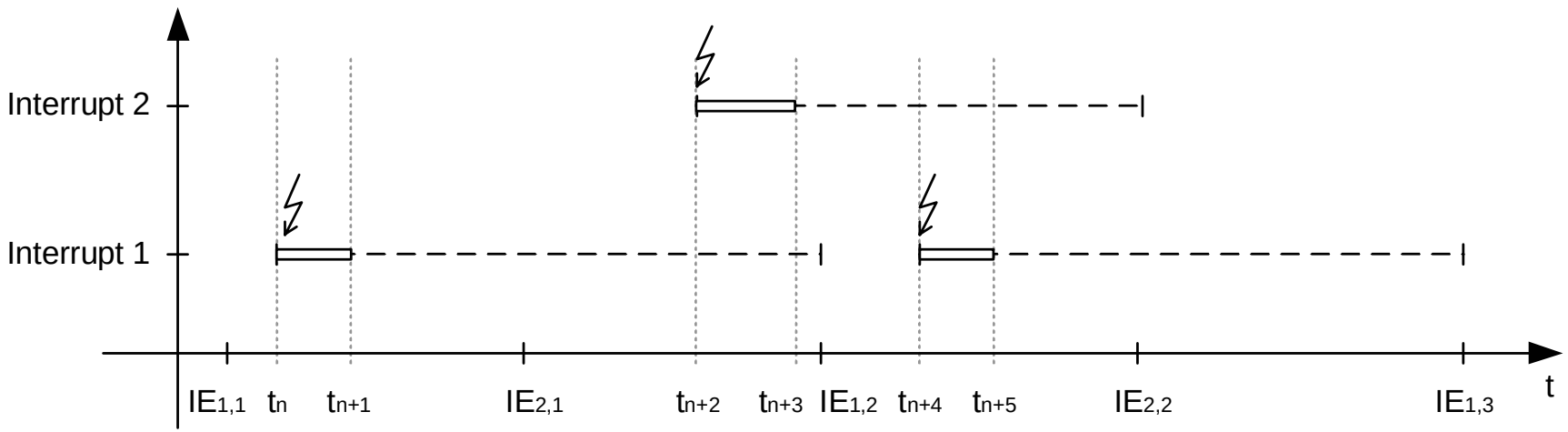

laufend
terminiere
aktiviere
suspendiert
wieder aufnehmen
verdränge
verdrängt

Deadlines

- **Strenge-Task-Deadline-Beobachtung**
  Beobachter wird direkt zum Zeitpunkt der Deadline aktiviert.
- **Keine-Strenge-Task-Deadline-Beobachtung**
  Beobachter wird an einer beliebigen Stelle, nach der Deadline aber vor dem Rundenende, positioniert.

Interrupt-Behandlung

- jeder Interrupt hat sein Intervall, in dem dieser nur maximal einmal behandelt wird
- Reaktivierungszeitpunkte werden in der Ablauftabelle definiert
- nichtmaskierbare Interrupts können zu unvorhersagbaren Verzögerungen führen
- keine Möglichkeit Interrupts manuell zu deaktivieren


Interrupt 2
Interrupt 1
IE1,1
tn
tn+1
IE2,1
tn+2
tn+3
IE1,2
tn+4
tn+5
IE2,2
IE1,3
t

Uhrensynchronisation

- jede ECU besitzt eine lokale Zeit
- bei Existenz einer globalen Zeit, muss die Lokale mit der Globalen synchronisiert werden
- 3 Synchronisationsarten:
  - Synchroner Start
  - Asynchroner Start - harte Synchronisation
  - Asynchroner Start - weiche Synchronisation

Überblick

Zusammenfassung

- OSEK / OSEKtime sind statische Betriebssysteme
- ereignisgesteuert vs. zeitgesteuert
- komplexe Tasks vs. einfache Tasks
- unterschiedliche Interrupt-Behandlung