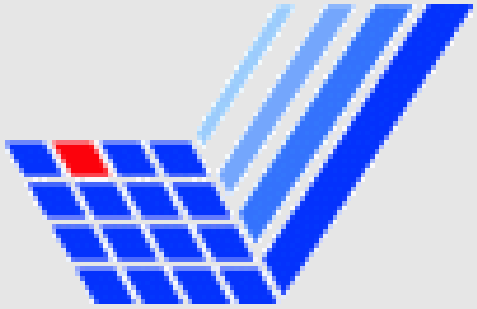
OSEK-OS

Oliver Botschkowski

oliver.botschkowski@udo.edu

PG AutoLab
Seminarwochenende 21.-23. Oktober 2007



Überblick

- Einleitung
  - Motivation
  - Ziele
  - Vorteile
- Einführung in OSEK-OS
  - Architektur
  - Task Management
  - Interrupt Processing
  - Event Mechanism
  - Resource Management
- Quellen
- Fragen

Thema

OSEK/VDX – OS

Offene **S**ysteme und deren Schnittstellen für die **E**lektronik im **K**raftfahrzeug

Vehicle **D**istributed e**X**ecutive

Initiale Partner

DAIMLERCHRYSLER

BOSCH

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

SIEMENS

Ca. 50 weitere Firmen die im Automotive Bereich tätig sind...

Einleitung

Einleitung - Motivation

- **Wiederkehrende Kosten** für die Entwicklung und Anpassung von Control Unit Software
- **Inkompatibilitäten** von Control Units die von verschiedenen Firmen hergestellt werden

Einleitung - Ziele

- Portabilität und Wiederverwendbarkeit von Anwendungssoftware gewährleisten durch:
  - **Abstrakte Interfaces** die so anwendungsunabhängig wie möglich gestaltet sind (Bereiche: Real Time OS, Kommunikation, Netzwerk Management)
    - Spezifikation eines **User Interfaces** unabhängig der Hardware und des Netzwerk
  - Effizientes Architektur-Design: **Konfigurierbar**, **Skalierbar**, um optimal auf die Anwendung abgestimmt zu werden
  - **Verifikation** der Funktionalitäten und Implementation von Prototypen in Pilot Projekten

Einleitung - Vorteile

- Einsparung von **Kosten** und **Entwicklungszeit**
- **Standardisiertes** Interface für verschiedene Control Units
- **Verbesserte Qualität** der Software von verschiedenen Control Units
- Spezifikation schreibt die Implementation nicht vor → **individuelle Implementierung** möglich

Einführung in OSEK-OS

OSEK-OS - Architektur

- OSEK-OS wird bei Erzeugung einmal **statisch konfiguriert** und **skaliert** (z.B. Anzahl an **Tasks** und benötigter **Services**)
- Real-Zeit Ausführung mehrerer **paralleler Prozesse**

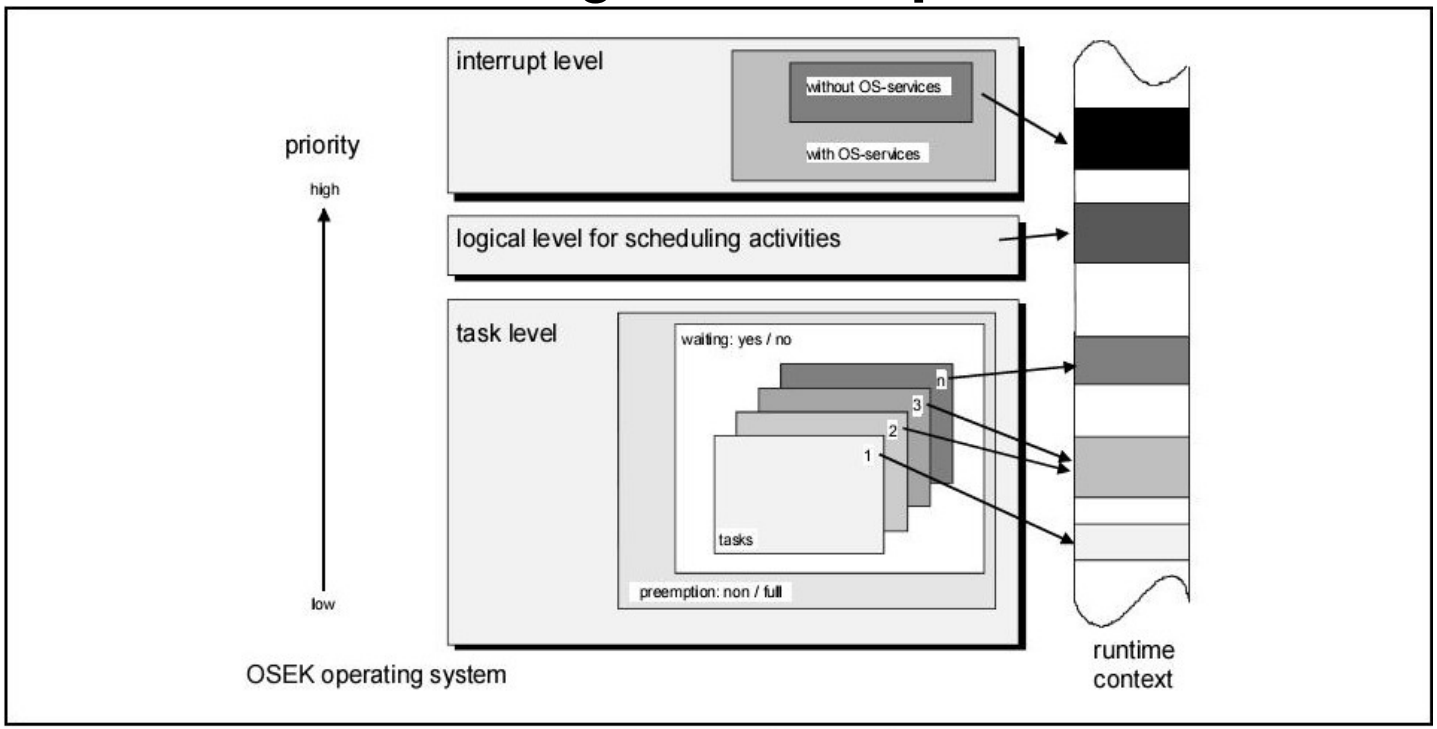


Figure 3-1 Processing levels of the OSEK operating system

OSEK-OS - Task Management

- **Basic Task**

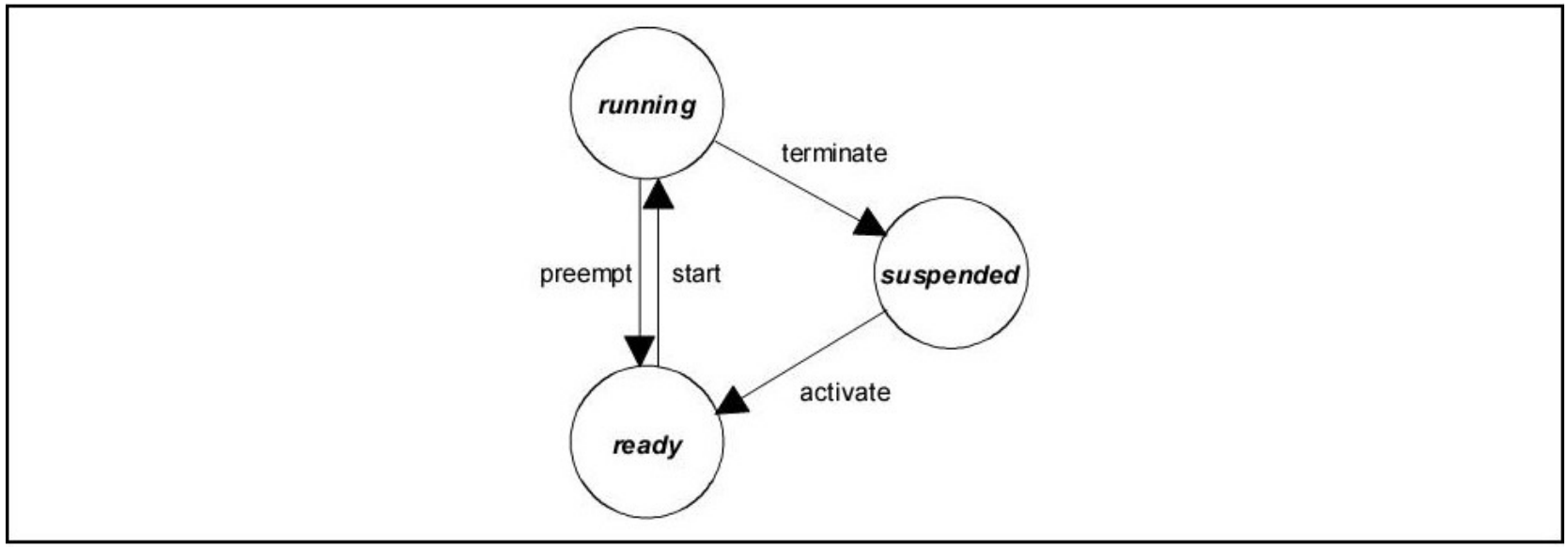


Figure 4-3 Basic task state model

OSEK-OS - Task Management

- **Prioritäten**

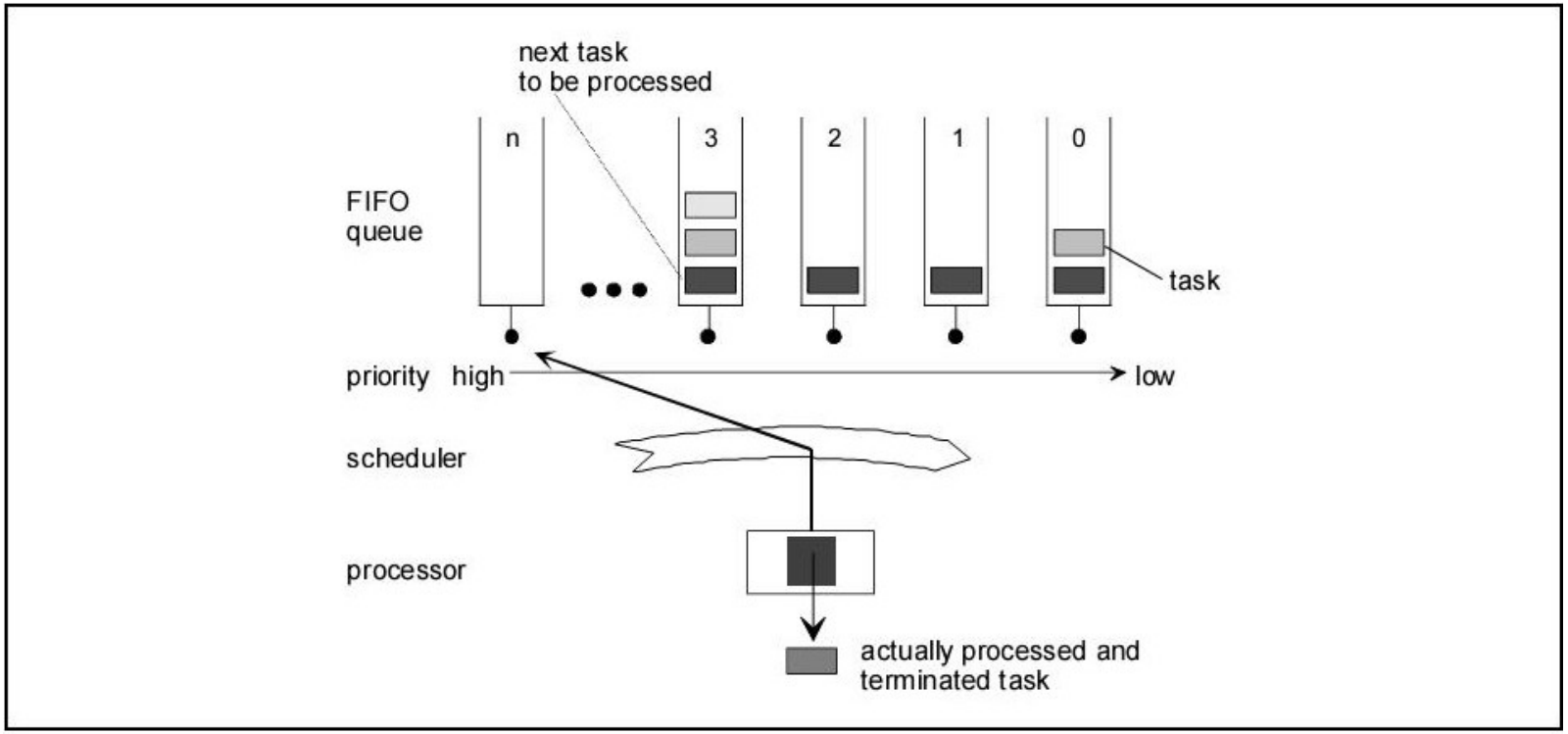


Figure 4-5 Scheduler: order of events

OSEK-OS - Task Management

- **Extended Tasks**

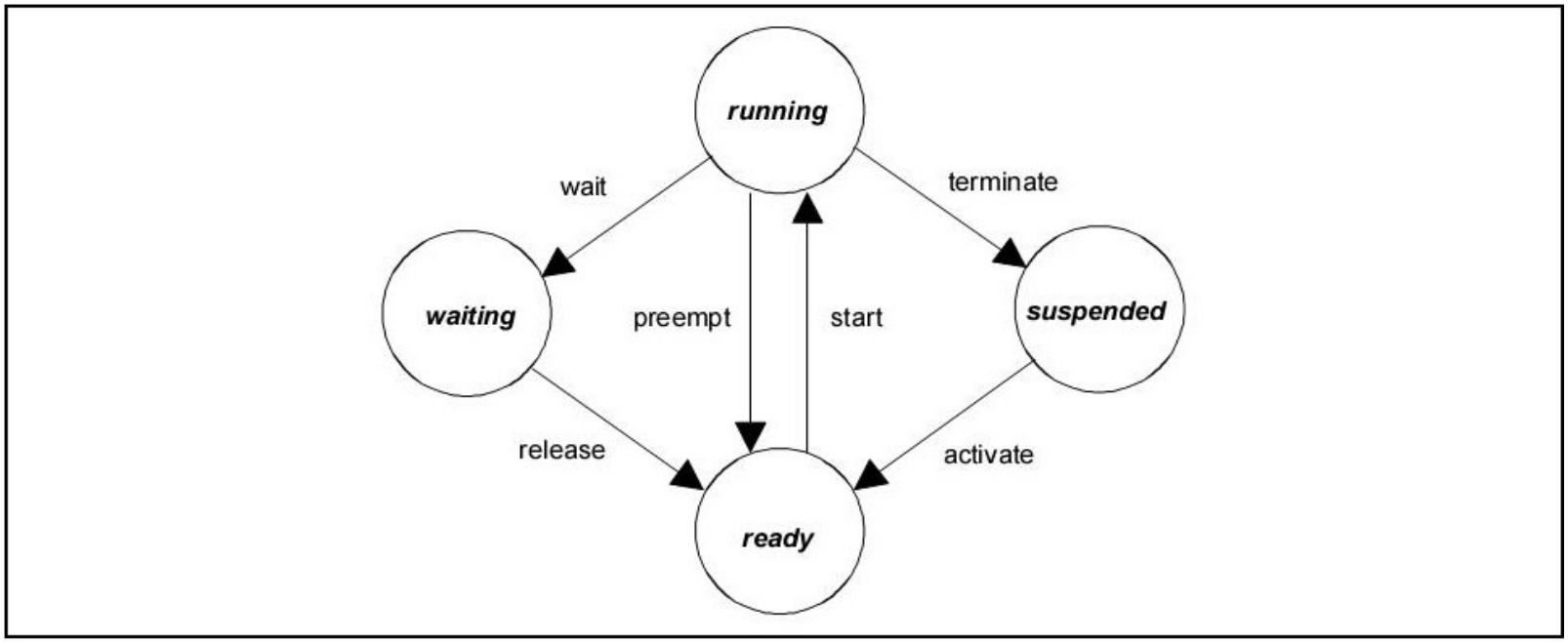


Figure 4-1 Extended task state model

OSEK-OS - Task Management

- **Full Preemptive Scheduling**
- Scheduling bei:
  - **Terminierung** eines Tasks
  - **Aktivierung** eines Tasks auf Task-Level
  - **Transition** in den Waiting State
  - **Event** bei einem wartenden Task
  - **Freigabe** von Ressourcen auf Task Level
  - **Rückkehr** von Interrupt auf Task Level

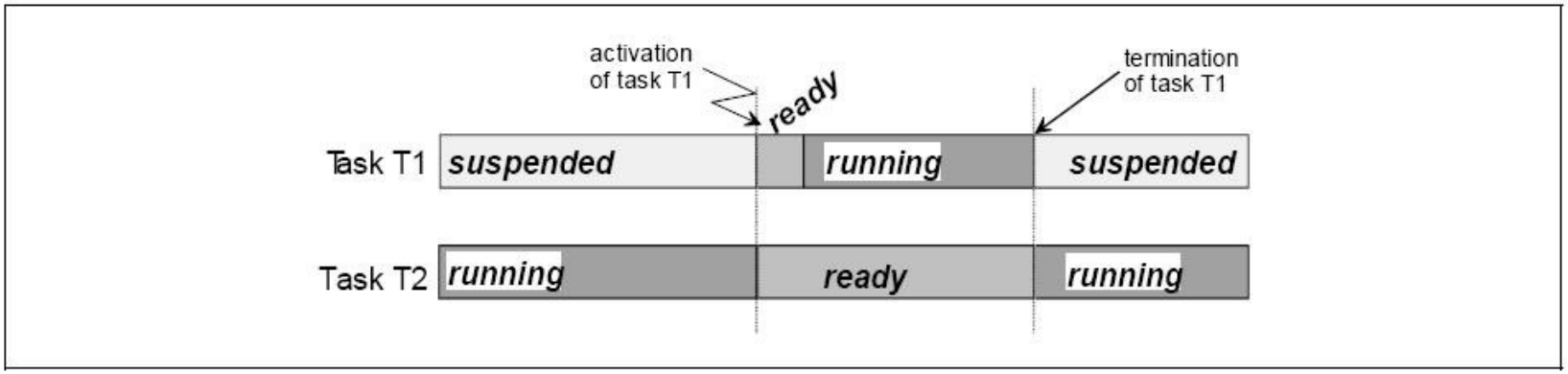


Figure 4-6 Full preemptive scheduling

OSEK-OS - Task Management

- **Non Preemptive Scheduling**
- Scheduling bei:
  - **Terminierung** eines Tasks
  - Expliziten **Aufruf** des Schedulers
  - **Transition** in den Waiting State

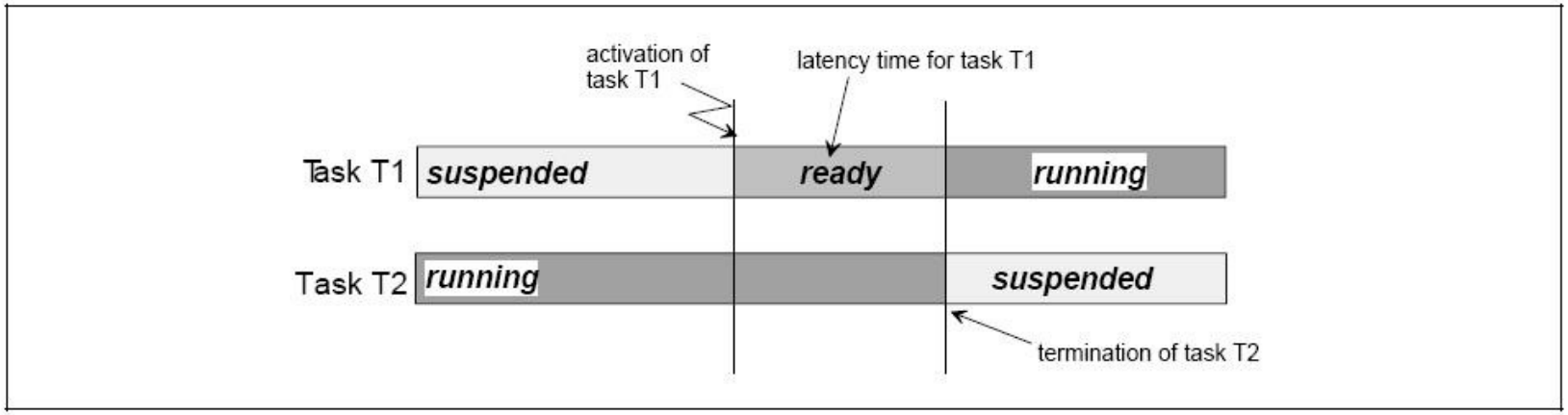


Figure 4-7 Non preemptive scheduling

OSEK-OS - Task Management

- **Mixed Preemptive Scheduling**
- Jeder Task kann preemptive oder non preemptive gescheduled werden

OSEK-OS – Interrupt Processing

- **ISR** (Interrupt Service Routine)
- Mehrere Interrupts auch mit Prioritäten
- Gescheduled durch die Hardware
- **Kategorie1**
  - ISR benutzt keine OS-Services
  - Unterbrochener Kontrollfluss wird fortgesetzt
  - Wenig Overhead
- **Kategorie2**
  - ISR kann einige OS-Services nutzen
  - Evtl. Aufruf des Schedulers

OSEK-OS - Event Mechanism

- **Events** = **Objekte** die durch das OS gehandhabt werden
- Gehören jeweils zu einem Extended Task
- Erwirkt **Zustandsübergänge** von Waiting zu Ready
- Werden zur **Kommunikation** zwischen Tasks genutzt

OSEK-OS - Event Mechanism

- Wie durch Events synchronisieren?
- T1 hat höhere Priorität als T2

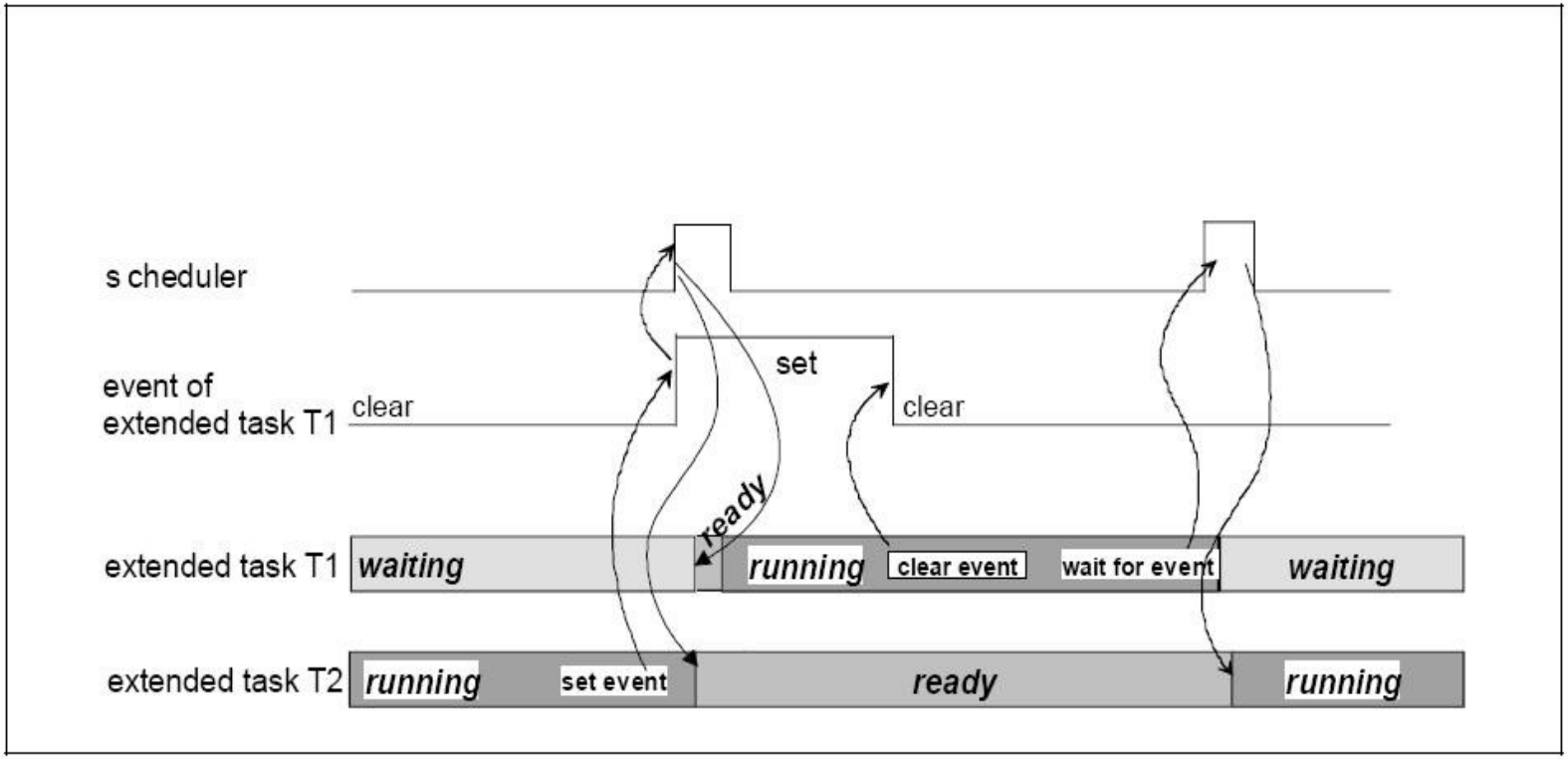


Figure 7-1 Synchronisation of preemptable extended tasks

OSEK-OS - Event Mechanism

- T1 hat höhere Priorität als T2

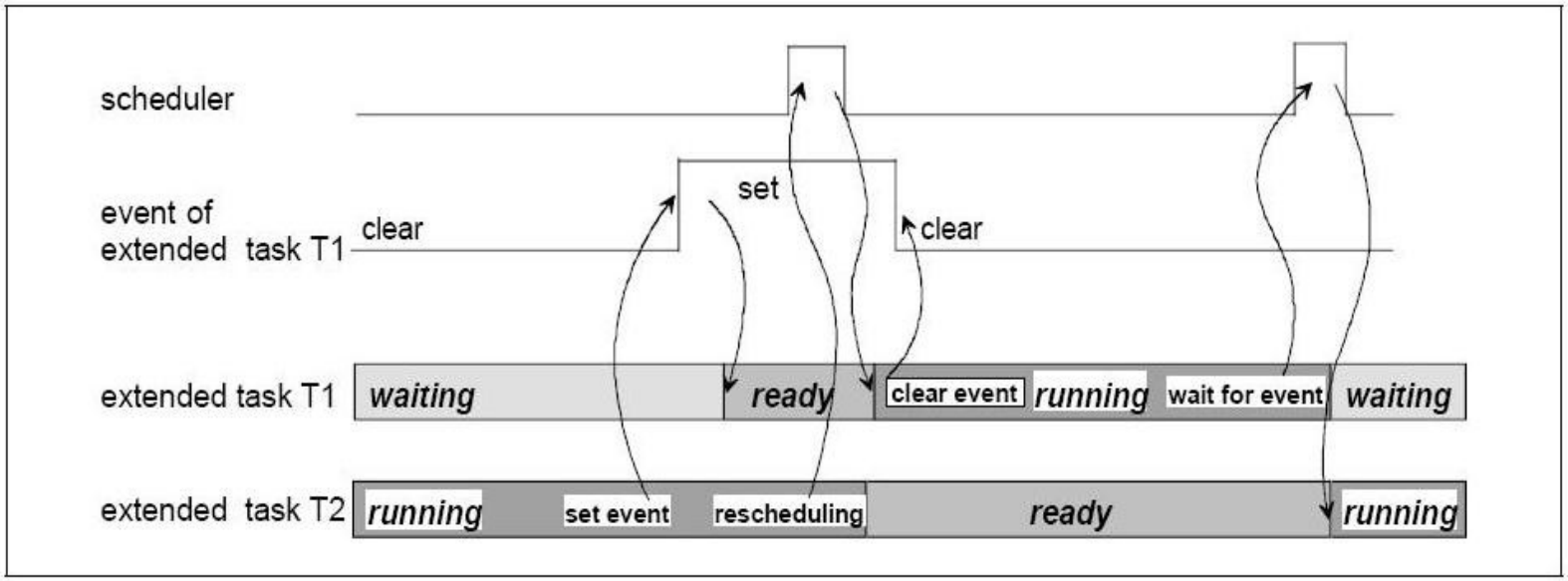


Figure 7-2 Synchronisation of non preemptable extended tasks

OSEK-OS - Resource Management

- **Koordination** konkurrierender Tasks beim Zugriff auf Resourcen
- Resource Management sichert folgendes zu:
  - Zwei Tasks können nicht die gleiche Resource zur selben Zeit haben
  - Keine **Priority Inversion**
  - Kein **Deadlock**
  - Der Zugriff auf Resourcen endet nicht im Waiting State
- Bei Erweiterung auf ISR:
  - Zwei Taks oder ISR´s können nicht die gleiche Resource zur selben Zeit nutzen

OSEK-OS - Resource Management

- Priority Ceiling Protocol

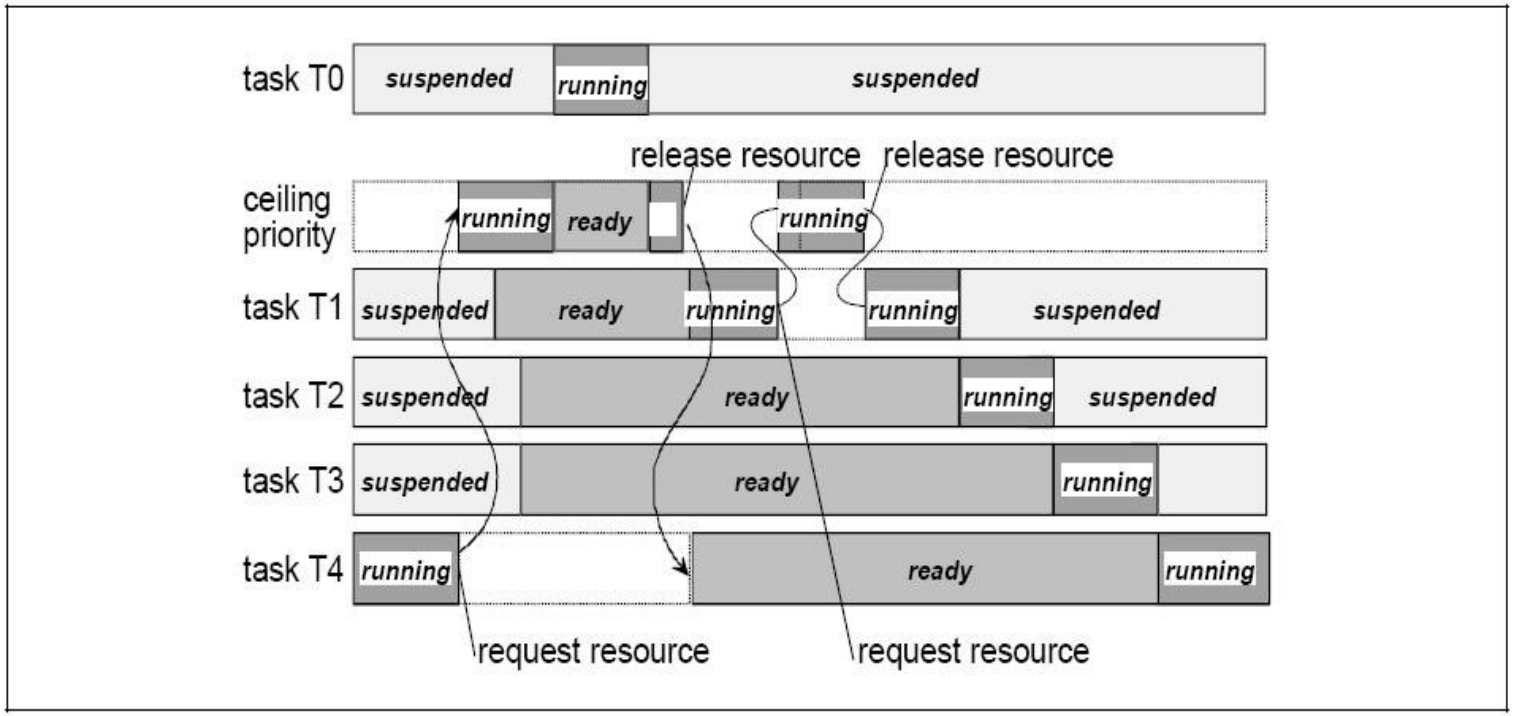


Figure 8-3 Resource assignment with priority ceiling between preemptable tasks.

OSEK-OS - Architektur

- **Skalierung** wird durch **Conformance Classes** abgesichert

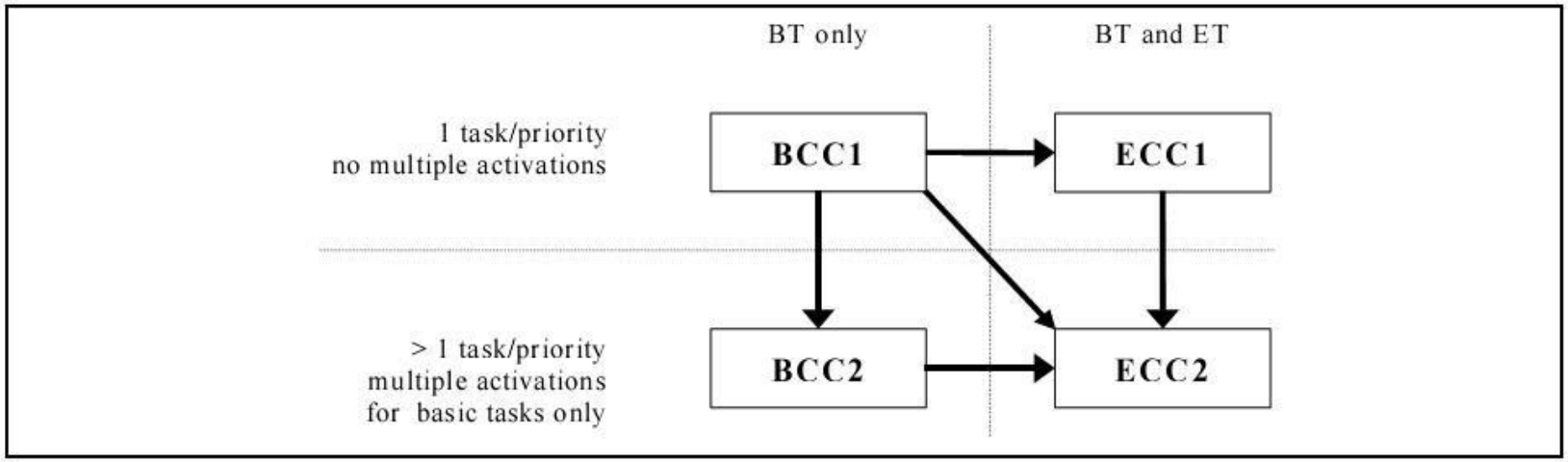


Figure 3-2 Restricted upward compatibility for conformance classes

Quellen

- www.osek-vdx.org
- OSEK/VDX Operating System Specification 2.2.3
- OSEK/VDX Binding Specification 1.4.2
- http://de.wikipedia.org/wiki/OSEK
- http://de.wikipedia.org/wiki/OSEK-OS

?Fragen?

Danke für eure Aufmerksamkeit!