



# Echtzeitsysteme

## Wintersemester 2009/2010

Prof. Dr. Alois Knoll, Dr. Christian Buckl

TU München

Lehrstuhl VI Robotics and Embedded Systems



# Echtzeitsysteme: Organisation

## Team



Prof. Dr. Alois Knoll

Dr. Christian Buckl



**Übungen:** Simon Barner, Irina Gaponova, Stephan Sommer

Homepage der Vorlesung mit Folien, Übungsaufgaben und weiterem Material:  
<http://www6.informatik.tu-muenchen.de/Main/TeachingWs2009Echtzeitsysteme>

**Frage:** Gibt es Hörer, denen eine englischsprachige Vorlesung bzw. Folien lieber wäre?



## Bestandteile der Vorlesung

- Vorlesung:
  - Dienstag 10:15-11:45 Uhr MI HS 2
  - Mittwoch 12:15-13:00 Uhr MI HS 2
  - 6 ECTS Punkte
  - Wahlpflichtvorlesung im Gebiet Echtzeitsysteme (Technische Informatik)
  - Wahlpflichtvorlesung für Studenten der Elektro- und Informationstechnik
  - Pflichtvorlesung für Studenten des Maschinenbau Richtung Mechatronik
- Übung:
  - zweistündige Tutorübung, im Raum 03.05.012
  - Mittwochs 14:00-15:30 Uhr
  - Mittwochs 15:30-17:00 Uhr
  - Weitere Termine bei Bedarf nach Vereinbarung
  - Beginn: voraussichtlich ab 04.11.2009
- Prüfung:
  - Schriftliche Klausur am Ende des Wintersemesters, falls ein Schein benötigt wird.



## Informationen zur Übung

- Ziel: Praktisches Einüben von Vorlesungsinhalten
- Übungsaufgaben werden in 2er Gruppen am Computer gelöst
- Platz ist begrenzt (7 Computer  $\Leftrightarrow$  14 Studenten)  $\Rightarrow$  **Anmeldung erforderlich**
- Übungsaufgaben sind auch auf der Vorlesungsseite verfügbar
- Es werden diverse Aufgaben aus dem Bereich der Echtzeitprogrammierung angeboten, wie z.B. Aufgaben zu Threads, Semaphore, Kommunikation
- Programmiersprache ist überwiegend C, zu Beginn der Übung wird eine kurze Einführung in C angeboten
- Die Anmeldung erfolgt über Email an [buckl@fortiss.org](mailto:buckl@fortiss.org)
- Die Übungsinhalte sind nicht direkt prüfungsrelevant, tragen aber stark zum Verständnis bei.

## Mögliche Übungsinhalte

- Modellierungssprachen/-werkzeuge (Esterel Studio, SCADE, Ptolemy, EasyLab, FTOS)
- Programmierung von Echtzeitsystemen (Programmiersprache C, Betriebssysteme VxWorks, PikeOS)
  - Nebenläufigkeit (Threads, Prozesse)
  - Interprozesskommunikation / Synchronisation: Semaphore, Signale, Nachrichtenwarteschlangen
  - Unterbrechungsbehandlung
- Kommunikation (Ethernet, CAN)
- Programmierung von Adhoc-Sensornetzwerken (TinyOS, ZigBee)
- Umsetzung von diversen Demonstratoren (Aufzug, Kugelfall, Murmelbahn)
- **Ihre Rückmeldung ist wichtig!!!**



## Klausur

- Für Studenten, die einen Schein benötigen, wird am Ende der Vorlesung eine schriftliche Klausur angeboten.
- Stoff der Klausur sind die Inhalte der Vorlesung.
- Die Inhalte der Übung sind nicht direkt prüfungsrelevant, tragen allerdings zum Verständnis des Prüfungstoffes bei.
- Voraussichtlicher Termin: letzte Vorlesungswoche (in Absprache mit den Studenten), vermutlich 09.02.2008 10:00-12:00 Uhr
- Voraussichtlich erlaubte Hilfsmittel: keine



## Grundsätzliches Konzept

- Themen werden aus verschiedenen Blickrichtungen beleuchtet:
  - Stand der Technik in der Industrie
  - Stand der Technik in der Wissenschaft
  - Existierende Werkzeuge
  - Wichtig: nicht die detaillierte Umsetzung, sondern die Konzepte sollen erlernt werden
- Praktische Aufgaben in der Vorlesung und der Übung und Analogien zum Alltag werden zur Verdeutlichung theoretischer Inhalte verwendet
- In jedem Kapitel werden die relevanten Literaturhinweise referenziert
- Zur Erfolgskontrolle werden Klausuraufgaben der letzten Jahre am Ende eines Kapitels diskutiert
- Wir freuen uns über Fragen, Verbesserungsvorschläge und Feedback



## Weitere Angebote des Lehrstuhls

- Weitere Vorlesungen: Robotik, Digitale Signalverarbeitung, Maschinelles Lernen und bioinspirierte Optimierung I&II, Sensor- und kamerageführte Roboter
- Praktika: Echtzeitsysteme, Roboterfußball, Industrieroboter, Neuronale Netze und Maschinelles Lernen, Bildverarbeitung, Signalverarbeitung
- Seminare: Sensornetzwerke, Modellierungswerkzeuge, Busprotokolle, Objekterkennung und Lernen, Neurocomputing,
- Diplomarbeiten / Masterarbeiten
- Systementwicklungsprojekte / Bachelorarbeiten
- Guided Research, Stud. Hilfskräfte
- Unser gesamtes Angebot finden Sie unter <http://wwwknoll.in.tum.de>

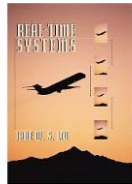
## fortiss – Münchner Software und Systeme Institut



- Spin-Off der TU München
- Gemeinnütziges Forschungsinstitut
- Gesellschafter:
  - Technische Universität München (Körperschaft des öffentlichen Rechts)
  - LfA, Förderbank Bayern
  - Fraunhofer Gesellschaft
- Gegründet im Januar 2009
- Ziel:
  - Aufbau eines Informatikinstitut im Großraum München
  - Schließen der Lücke zwischen Forschung an Universitäten und Firmen
  - Inkubator für Neugründungen

# Literatur

Hermann Kopetz: Real-Time Systems (Überblick)



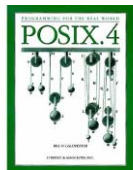
Jane W. S. Liu: Real-Time Systems  
(Überblick, Schwerpunkt Scheduling)

Stuart Bennet: Real-Time Computer Control:  
An Introduction (Überblick, Hardware)



Alan Burns, Andy Wellings: Real-Time Systems and Programming  
Languages (Schwerpunkt: Programmiersprachen)

Qing Li, Caroline Yao: Real-Time Concepts for  
Embedded Systems (Schwerpunkt: Programmierung)



Bill O. Gallmeister: Programming for the Real-World:  
POSIX.4 (Schwerpunkt: Posix)

Weitere Literaturangaben befinden sich in den jeweiligen Abschnitten.



## Vorlesungsinhalte

1. Einführung Echtzeitsysteme
2. Modellierung und Werkzeuge
3. Nebenläufigkeit
4. Scheduling
5. Echtzeitbetriebssysteme
6. Programmiersprachen
7. Uhren
8. Kommunikation
9. Fehlertolerante Systeme
10. Spezielle Hardware
11. Regelungstechnik

*Weitere Themen können bei Interesse aufgenommen werden. Melden Sie sich einfach nach der Vorlesung oder per Email.*

## Inhalt I

- Kapitel Einführung (ca. 1 Vorlesungswoche)
  - Definition Echtzeitsysteme
  - Klassifikation
  - Echtzeitsysteme im täglichen Einsatz
  - Beispielanwendungen am Lehrstuhl
- Kapitel Modellierung/Werkzeuge (ca. 3 Vorlesungswochen)
  - Allgemeine Einführung
  - Grundsätzlicher Aufbau, Models of Computation, Ptolemy
  - Synchrone Sprachen (Esterel, Lustre), SCADE, EasyLab
  - Zeitgesteuerte Systeme: Giotto, FTOS, TTA
  - Exkurs: Formale Methoden



## Inhalt II

- Kapitel Nebenläufigkeit (2 Vorlesungswochen)
  - Prozesse, Threads
  - Interprozesskommunikation
- Kapitel Scheduling (2 Vorlesungswochen)
  - Kriterien
  - Planung Einrechner-System, Mehrrechnersysteme
  - EDF, Least Slack Time
  - Scheduling mit Prioritäten (FIFO, Round Robin)
  - Scheduling periodischer Prozesse
  - Scheduling Probleme



## Inhalt III

- Kapitel Echtzeitbetriebssysteme (1 Vorlesungswoche)
  - QNX, VxWorks, PikeOS
  - RTLinux, RTAI, Linux Kernel 2.6
  - TinyOS, eCos
  - OSEK
- Kapitel Programmiersprachen (1 Vorlesungswoche)
  - Ada
  - Erlang
  - C mit POSIX.4
  - Real-time Java
- Kapitel Uhren (1 Vorlesungswoche)
  - Uhren
  - Synchronisation von verteilten Uhren



## Inhalt IV

- Kapitel Echtzeitfähige Kommunikation (1 Vorlesungswoche)
  - Token-Ring
  - CAN-Bus
  - TTP, FlexRay
  - Real-Time Ethernet
- Kapitel Fehlertoleranz (ca. 2 Vorlesungswochen)
  - Bekannte Softwarefehler
  - Definitionen
  - Fehlerarten
  - Fehlerhypothesen
  - Fehlervermeidung
  - Fehlertoleranzmechanismen



## Potentielle zusätzliche Inhalte

- Kapitel: Spezielle Hardware (1 Vorlesungswoche)
  - Digital-Analog-Converter (DAC)
  - Analog-Digital-Converter (ADC)
  - Speicherprogrammierbare Steuerung (SPS)
- Kapitel: Regelungstechnik (ca. 2 Vorlesungswochen)
  - Definitionen
  - P-Regler
  - PI-Regler
  - PID-Regler
  - Fuzzy-Logic

## Guided Research / Werkstudentenstellen

- Themen:
    - Modellbasierte Entwicklungswerkzeuge
      - Modellbasierte Entwicklung von verteilten (fehlertoleranten) Echtzeitsystemen (Werkzeuge: Eclipse, openArchitectureWare), Projekte (BMBF-Projekte SEIS, ePerformance, Multifunk)
    - Parallele Architekturen:
      - Programmierung von FPGAs und Multicore-Architekturen (EU-Projekte: SMECY\*, ReComp\*)
      - Entwurf deterministischer Multicore-Architekturen (EU-Projekt: ACROSS\*)
      - Entwurf modellbasierter Entwicklungswerkzeuge und geeigneter Middleware-Architekturen
    - Kombination von formalen Methoden
      - Formale Methoden in der Entwicklung am Beispiel vom Elektroauto (Anbindung in Graduiertenkolleg PUMA)
    - Entwicklung von Demonstratoren
      - Elektroauto (BMBF-Projekte SEIS, ePerformance)
      - Produktionsstraße (BMBF-Projekt Multifunk)
    - Weitere Themen auf Nachfrage bzw. auf der Lehrstuhlhomepage
  - Ziel:
    - Enge Anbindung von interessierten Studenten an den Lehrstuhl/fortiss
    - Jeder Student bekommt einen „Mentor“ inkl. Arbeitsplatz zur Bearbeitung der Themen
  - Bei Interesse senden Sie bitte Ihre Bewerbungsunterlagen an Christian Buckl
- \* Entscheidung über Förderung Ende Oktober*



# Kapitel 1

## Einführung Echtzeitsysteme



## Inhalt

- Definition Echtzeitsysteme
- Klassifikation von Echtzeitsystemen
- Echtzeitsysteme im täglichen Leben
- Beispielanwendungen am Lehrstuhl

## Definition Echtzeitsystem

Ein Echtzeit-Computersystem ist ein Computersystem, in dem die Korrektheit des Systems nicht nur vom logischen Ergebnis der Berechnung abhängt, sondern auch vom physikalischen Moment, in dem das Ergebnis produziert wird.

Ein Echtzeit-Computer-System ist immer nur ein Teil eines größeren Systems, dieses größere System wird Echtzeit-System genannt.

*Hermann Kopetz*

*TU Wien*

## Definition Eingebettetes System

Technisches System, das durch ein integriertes, von Software gesteuertes Rechensystem gesteuert wird. Das Rechensystem selbst ist meist nicht sichtbar und kann in der Regel nicht frei programmiert werden. Um die Steuerung zu ermöglichen ist zumeist eine Vielzahl von sehr speziellen Schnittstellen notwendig.

In der Regel werden leistungsärmere Mikroprozessoren mit starken Einschränkungen in Bezug auf die Rechenleistung und Speicherfähigkeit eingesetzt.

## Resultierende Eigenschaften

⇒ zeitliche Anforderungen

- Zeitliche Genauigkeit (nicht zu früh, nicht zu spät)
- Garantierte Antwortzeiten
- Synchronisation von Ereignissen / Daten
- **Aber nicht:** Allgemeine Geschwindigkeit

⇒ Eigenschaften aufgrund der Einbettung

- Echtzeitsysteme sind typischerweise sehr Eingabe/Ausgabe (E/A)-lastig
- Echtzeitsysteme müssen fehlertolerant sein, da sie die Umgebung beeinflussen
- Echtzeitsysteme sind häufig verteilt

## Zeitlicher Determinismus vs. Leistung

- Konsequenz der Forderung nach deterministischer Ausführungszeit: Mechanismen, die die allgemeine Performance steigern, aber einen negativen, nicht exakt vorhersehbaren Effekt auf einzelne Prozesse haben können, werden in der Regel nicht verwendet:
  - Virtual Memory
  - Garbage Collection
  - Asynchrone IO-Zugriffe
  - rekursive Funktionsaufrufe



## Klassifikation von Echtzeitsystemen

- Echtzeitsysteme können in verschiedene Klassen unterteilt werden:
  - Nach den Konsequenzen bei der Überschreitung von Fristen: harte vs. weiche Echtzeitsysteme
  - Nach dem Ausführungsmodell: zeitgesteuert (zyklisch, periodisch) vs. ereignisbasiert (aperiodisch)

## Harte bzw. weiche Echtzeitsysteme

- **Weiche Echtzeitsysteme:**

Die Berechnungen haben eine zeitliche Ausführungsfrist, eine Überschreitung dieser Fristen hat jedoch keine katastrophale Folgen. Eventuell können die Ergebnisse noch verwendet werden, insgesamt kommt es durch die Fristverletzung evtl. zu einer Dienstverschlechterung.

**Beispiel für ein weiches Echtzeitsystem:** Video

**Konsequenz von Fristverletzungen:** einzelne Videoframes gehen verloren, das Video hängt

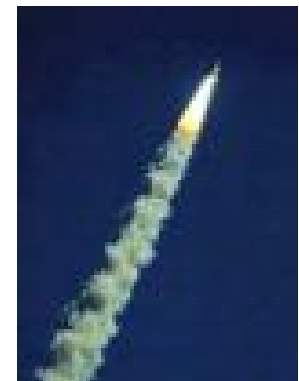


- **Harte Echtzeitsysteme:**

Eine Verletzung der Berechnungsfristen kann sofort zu fatalen Folgen (hohe Sachschäden oder sogar Gefährdung von Menschenleben) führen. Die Einhaltung der Fristen ist absolut notwendig.

**Beispiel für ein hartes Echtzeitsystem:** Raketensteuerung

**Konsequenz von Fristverletzung:** Absturz bzw. Selbstzerstörung der Rakete



## Unterteilung nach Ausführungsmodell

- Zeitgesteuerte Applikationen:
  - Der gesamte zeitliche Systemablauf wird zur Übersetzungszeit festgelegt
  - Notwendigkeit einer präzisen, globalen Uhr  $\Rightarrow$  Uhrensynchronisation notwendig
  - Für die einzelnen Berechnungen ist jeweils ein Zeitslot reserviert  $\Rightarrow$  Abschätzung der maximalen Laufzeiten (**worst case execution times - WCET**) notwendig
  - **Vorteil:** Statisches Scheduling möglich und damit ein vorhersagbares (**deterministisches**) Verhalten
- Ereignisgesteuerte Applikationen:
  - Alle Ausführungen werden durch das Eintreten von Ereignissen angestoßen
  - Wichtig sind bei ereignisgesteuerten Anwendungen garantierte Antwortzeiten
  - Das Scheduling erfolgt dynamisch, da zur Übersetzungszeit keine Aussage über den zeitlichen Ablauf getroffen werden kann.