

# Wiederholung

## Nicht-periodische Aufgaben

Tobias Klaus Florian Schmaus Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)  
Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme)  
<https://www4.cs.fau.de>

9. Januar 2017



# Übersicht

1 Wiederholung: Nicht-periodische Aufgaben

2 Interrupts in Echtzeitsystemen

3 Aufgabe 6: Extended Scope



## Rekapitulation der Vorlesung

Kapitel 5-1: Grundlegende Abfertigung nicht-periodischer Echtzeitsysteme

### Nicht-periodische Aufgaben

- Definiert durch  $T_i = (i_i, e_i, D_i)$
- *Aperiodische* vs. *sporadische* Aufgabe
- *Mischbetrieb*: periodisch  $\leftrightarrow$  sporadisch/aperiodisch
  - *dynamische* Einplanung
  - Beeinflussung periodischer Aufgaben?
  - Übernahmeprüfung  $\leftrightarrow$  Antwortzeitminimierung

### Nicht-periodische Arbeitsaufträge

- Kaum a-priori Wissen (Zeitpunkt, WCET, ...)
- Herausforderung Mischbetrieb: Erhaltung statischer Garantien
- Abweisung (spor. Aufg.): Schwerwiegende Ausnahmesituation



## Rekapitulation der Vorlesung (Forts.)

Kapitel 5-1: Grundlegende Abfertigung nicht-periodischer Echtzeitsysteme

### Basistechniken zur Umsetzung

- *Unterbrecherbetrieb*  $\leadsto$  Bevorzugt nicht-periodische Aufgaben
- *Hintergrundbetrieb*  $\leadsto$  Stellt nicht-periodische Aufgaben hinten an
- *Slack Stealing*
  - Idee: Termin ist maßgeblich  
 $\leadsto$  *Verschieben* periodischer Aufgaben möglich
  - *Erfordert Unterbrecherbetrieb*
  - Problem: Schlupfzeit bestimmen
    - Zeitsteuerung (mit Rahmen): Einfach  $\leadsto f - x_k$
    - Ereignissteuerung: Schwierig  $\leadsto$  dynamischen Berechnung
- *Zusteller*  $\leadsto$  Konvertieren nicht-period. in periodische Aufgaben
  - Spezielle periodische Aufgabe  $T_s = (p_s, e_s)$
  - Ausführungsbudget, Auffüllperiode und -regeln
  - Abbildung auf Prioritätswarteschlange (z. B. AJQ)



## Rekapitulation der Vorlesung (Forts.)

### Kapitel 5-1: Grundlegende Abfertigung nicht-periodischer Echtzeitsysteme

#### Periodische Zusteller

- Verschiedene Ausführungen  
z. B.: Polling, Deferrable, Sporadic Server
- Unterscheiden sich im Regelwerk
- I. d. R. für mehrere Aufgaben zuständig

#### Beispiel: Abfragender Zusteller (Polling Server)

- Periodische Aufgabe  $T_P = (p_s, e_s)$
- Budget  $e_s$  verfällt
- Im Falle sporadischer Aufgaben schwierig:
  - $p_P \leq \frac{D_s}{2}$ , wobei  $D_s \leq i_s \leadsto$  Abtasttheorem
  - hohe Abtastfrequenz, Überlastgefahr



## Übersicht

1 Wiederholung: Nicht-periodische Aufgaben

2 Interrupts in Echtzeitsystemen

3 Aufgabe 6: Extended Scope

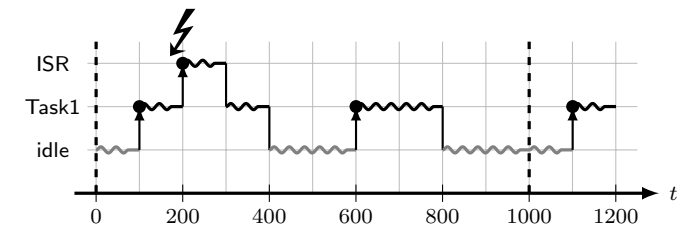


## Begriffe

- *Interrupt*
- *Interrupt-Vektor*
- *Interrupt-Handler*
- *Interrupt-Controller*
- *Pending*
- *Interrupt-Latenz*
- *Geschachtelter Interrupt*



## Probleme von Interrupts [1, 2]



- *Prioritätsverletzungen*
- *Prellen*
  - Entprellung in Soft- oder Hardware
  - Tiefpassfilterung
- *Auftrittshäufigkeit*
  - Maximale Auftrittsfrequenz (= minimale Zwischenankunftszeit)
  - Soft- oder Hardware-seitige Überwachung
- *Auftrittszeitpunkte*
  - Zeitliche Garantien jederzeit gewährleisten
  - Zugriff auf Ressourcen



## Checkliste für Interrupts in (V)EZS [1]

- 1 **Scheduling:** Analyse muss Interrupts (als Overheads) beachten
- 2 **Aufrufgraphen**
  - Identifikation der Kontexte in denen Interrupts auftreten könnten
  - ISR → DSR → `cyg_thread_resume()`
- 3 **Zeitanalyse**
  - Bestimmung der maximalen Auftrettsfrequenz
  - WCET-Analyse der Interrupts
- 4 **Korrektheit des Stacks**
  - Effekte von Interrupts auf Stack-Anordnungen
  - Bestimmung von Stack-Budgets (worst-case stack usage)
- 5 **Korrektheit der Nebenläufigkeiten**
  - Identifikation von Datenstrukturen auf die nebenläufig zugegriffen wird
  - Vermeidung von Race-Conditions, Verwendung atomarer Operationen



## Interrupt Scheduler [2]

- Detektion und Behandlung falscher Interrupts (engl. spurious interrupts)
- Externe Geräte können fehlerhaft sein  $\leadsto$  *Babbling Idiot*
- Software-Lösung (weiche Echtzeit)
  - Zählen von Interrupts über Zeitintervall
  - Linux <sup>1</sup>
  - Nur möglich wenn  $WCET(ISR) < \frac{1}{\text{minimale Zwischenankunftszeit}}$ 
    - Detektion von spurious Interrupts
    - Deaktivierung des IRQs
    - Adaptives Pollen von Geräten
    - Zusätzlicher Laufzeit-Overhead
- Hardware-Lösung (harte Echtzeit)
  - Zählen der Auftrettsfrequenzen in Hardware
  - TriCore CPU erlaubt das Zählen von externen Ereignissen (Komparatoren)
  - Überwachung implementierbar
  - Kein zusätzlicher Laufzeit-Overhead (außer Konfigurationsaufwand)

<sup>1</sup><https://github.com/torvalds/linux/blob/master/kernel/irq/spurious.c>



## Übersicht

- 1 Wiederholung: Nicht-periodische Aufgaben
- 2 Interrupts in Echtzeitsystemen
- 3 Aufgabe 6: Extended Scope



## Aufgabe 6: Extended Scope

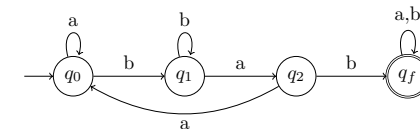


Abbildung: Beispiel für Zustandsmaschine

- Befehlsschnittstelle für Oszilloskop
- Auswertung von Benutzereingaben
  - *Was ist hier problematisch?*
  - Unterbrecher-, Hintergrundbetrieb, Periodischer Zusteller
- **Moduswechsel**
  - Dynamische Anpassung auf Last (Rekonfiguration der Ablauf tabellen)
  - Siehe Vorlesung 4-3
  - Systemweite Koordination mittels *Zustandsmaschine*
- Erweiterte Übung
  - *Ausgabe & Besprechung nächste Woche*
  - Rangfolge
  - Mailboxen, Events



[1] John Regehr.

Safe and structured use of interrupts in real-time and embedded software.

*Handbook of Real-Time and Embedded Systems*, I. Lee, JY-T. Leug, and SH Son, Eds. Chapman and Hall/CRC, pages 1–13, 2007.

[2] John Regehr and Usit Duongsaa.

Preventing interrupt overload.

*In Proceedings of the 2005 ACM SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems (LCTES '05)*, pages 50–58, New York, NY, USA, 2005. ACM Press.

