

# Grundlegende Übungen

## Aufgabe 1 - zeitgesteuerte Ablaufplanung

Fabian Scheler, Peter Ulbrich, Niko Böhm

5. November 2009

Ausgangspunkt für diese Aufgabenstellung ist ein rein zeitgesteuerter Ablaufplaner. Der Ablaufplaner hat dabei die Eigenschaften, wie sie in den Tafelübungen besprochen wurden. Gegeben sei weiterhin eine Menge von Aufgaben:

| Aufgabe | p/a | Periode | min. Zwischenankunftszeit | WCET |
|---------|-----|---------|---------------------------|------|
| T1      | p   | 20 ms   | NA                        | 4 ms |
| T2      | p   | 40 ms   | NA                        | 5 ms |
| T3      | p   | 80 ms   | NA                        | 2 ms |
| T4      | p   | 30 ms   | NA                        | 1 ms |
| T5      | a   | NA      | 120 ms                    | 3 ms |

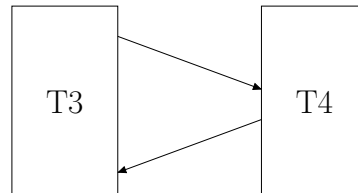
**Aufgabe 1** Implementieren Sie mindestens 2 verschiedene Ablauftabellen, in denen die Aufgaben T1, T2, T3 und T4 abgearbeitet werden.

- Erläutern Sie die Möglichkeiten die es gibt, die Aufgaben aufzuteilen  
**Hinweis:** Kapitel 3 - Grundlagen, Folie 3-10.
- Erläutern und zeigen Sie mit Hilfe einer Messung die Vor- und Nachteile der verschiedenen Varianten. **Hinweis:** Es hat etwas mit den Startzeitpunkten der Aufgaben zu tun.
- Unter welchen Voraussetzungen sind die verschiedenen Varianten *gleich gut*? **Hinweis:** Wie genau weiß man über die Ausführungszeit wirklich Bescheid?

**Aufgabe 2** Nehmen Sie an, die WCET der Aufgabe T3 betrage nun nicht mehr 2 ms, sondern 30 ms. Implementieren Sie eine Ablauftabelle, die dies berücksichtigt.

- Die vorliegende Anwendung ist sehr einfach - reale Anwendungen haben eine wesentlich höhere Komplexität. Welche Konsequenz könnte das Vorgehen haben, das Sie für diese Aufgabe angewandt haben?  
**Hinweis:** Wie sieht es mit der Verwendung lokaler Variablen aus?
- Wie könnte ein alternativer Ansatz aussehen, der keine Probleme mit lokalen Variablen hat? **Hinweis:** Es gibt Ähnlichkeiten zu Unterbrechungen (engl. *Interrupts*). Diese Variante kann mit dem vorliegenden EZStubs nicht implementiert werden.

**Aufgabe 3** Nehmen Sie an, die Aufgabe T4 habe ebenfalls eine Periode von 80 ms. Die Aufgaben T3 und T4 *kommunizieren* dabei noch folgendem Muster:



Aufgabe T3 berechnet also zunächst ein Ergebnis, das von T4 verwendet wird. T4 wiederum berechnet seinerseits ein Ergebnis, das von T3 verwendet wird. Implementieren sie eine Ablaufabelle, die dies berücksichtigt.

- Kann man hier einen ähnlichen Ansatz wie in Aufgabe 2 b) verwenden?
- Wie müsste man das Kommunikationsverhalten von T3 und T4 erweitern, dass man diesen Ansatz nicht mehr verwenden kann?

**Aufgabe 4** Implementieren Sie eine Ablaufabelle, die alle Aufgaben T1 - T5 abarbeitet. **Hinweis:** Simulieren Sie das aperiodische Ereignis mit Hilfe eines `Interrupt_Timers` (siehe EZStubs Einführung, Folien 27 - 29 und 41), den Sie mit einem zufälligen Wert aufziehen.

- Auf welche Probleme stoßen Sie bei der Behandlung der Aufgabe T5?
- Veranschaulichen Sie diese Probleme mit Hilfe einer Messung. **Hinweis:** Wieviel Zeit vergeht zwischen dem Auftreten des Ereignisses und dem Fertigstellen der Behandlung?

## Allgemeine Hinweise

- Erstellen Sie im Unterverzeichnis `tests` der Vorgabe ein weiteres Unterverzeichnis `gue1`. Erstellen sie dort für jede der Aufgaben eine Datei `aufgabeX.cc`  $X=1,2,3,4$ , in der Sie die einzelnen Aufgaben implementieren.
- Beantworten Sie die Zusatzfragen innerhalb der Dateien `aufgabeX.cpp`. Geben Sie jeweils deutlich an, auf welche Zusatzaufgabe sich Ihre Erläuterungen beziehen.
- Für etwaige Zeitmessungen verwenden Sie die Klasse `Timer` ((siehe EZStubs Einführung, Folien 27 - 29).
- In manchen Fällen sind Zufallszahlen hilfreich - in `debug/random.h` findet ihr die Schnittstelle eines Generators für Pseudozufallszahlen (Mersenne-Twister). **Wichtig:** Vor der Verwendung mit `random_init()` initialisieren.