

# Echtzeitsysteme

Übungen zur Vorlesung

Nachlese

**Simon Schuster**   **Peter Wägemann**

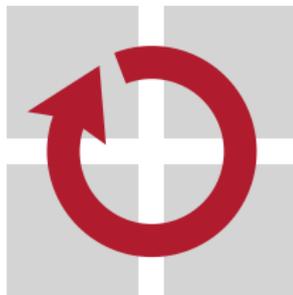
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)  
Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme)  
<https://www4.cs.fau.de>

Sommersemester 2022



- 1 Werbung
- 2 Rekapitulation
- 3 Fragen zur Klausur
- 4 Klausurvorbereitung





- Studienarbeiten im Bereich Echtzeitsysteme
  - Bachelor
  - Master
  - ...
- Weitere Lehrveranstaltungen
  - VEZS
  - DIY
  - ...



- 1 Werbung
- 2 Rekapitulation**
- 3 Fragen zur Klausur
- 4 Klausurvorbereitung



## AUFGABE 1: HALLO WELT

Diese Aufgabe dient dem initialen Einrichten Ihrer Entwicklungsumgebung und dem ersten Kontakt mit dem Echtzeitbetriebssystem eCos und dem STM32F429-Board. Ziel ist es einen ersten Einblick in die Möglichkeiten der Entwicklungsumgebung zu erhalten. Im Laufe der Übungen werden Sie mit vielen verschiedenen Themenbereichen in Berührung kommen: Programmierung in C, Entwicklung auf eingebetteten Systemen, Interaktion mit der Außenwelt und zeitliche Aspekte von Echtzeitsystemen. Dabei stellt die Programmierung ein notwendiges Mittel zum Zweck dar und wird als Vorkenntnis vorausgesetzt. Gegen Ende der Veranstaltung werden die zeitlichen Aspekte zunehmend mehr betont. In diesem Aufgabenblatt werden alle genannten Aspekte kurz thematisiert.

*Hinweis: Zeitliche Aspekte von Ein-/Ausgabevorgängen sind Bestandteil einer folgenden Tafelübung.*

### 1 Aufgabenstellung

Nach dem erfolgreichen Aufsetzen des build-Verzeichnisses können Sie sich mittels `nake doc` eine Übersicht über alle in libE2S bereitgestellten Funktionen inklusive deren Dokumentation erzeugen.

Denken Sie daran, Ihren Quellcode nach vollständiger Bearbeitung noch vor dem Beginn der Rechnerübung abzugeben. Rufen Sie hierzu in Ihrem build-Verzeichnis `nake submit` auf.

#### 1.1 Vorbereitung:

##### Aufgabe 1

Kopieren und entpacken Sie die Vorgabe in ein beliebiges Arbeitsverzeichnis und setzen Sie die nötigen Umgebungsvariablen durch den Aufruf von `source ecosenv.sh`. Nun können Sie die Makefiles generieren und die noch funktionslose Anwendung erstmals kompilieren. Sie können die erzeugte Anwendung mit Hilfe des Debuggers in den Flash-Speicher des Boards laden und starten.

*Hinweis: Zum Starten des Programms muss der schwarze Reset-Taster gedrückt werden!*

- C-Programmierung
- eCos & prozessorientierte Ausführungsmodelle
- Hardware-Interaktion, Polling
- Periodizität und aktives Warten



## AUFGABE 2: ANTWORTZEIT

In den ersten beiden Teilaufgaben dieser Aufgabe sollen Sie nützliche Funktionen für den Umgang mit Zeit in einem Rechesystem implementieren, die wir im weiteren Verlauf der Übung immer wieder verwenden werden. Die letzten Teilaufgaben sollen Ihnen dann ein erstes Gefühl für die Probleme beim Zusammenspiel von periodischen und nicht-periodischen Ereignissen in einem Echtzeitsystem vermitteln.

### 1 Aufgabenstellung

Nach dem erfolgreichen Aufsetzen des `build`-Verzeichnisses können Sie sich mittels `make doc` eine Übersicht über alle in `libEZS` bereitgestellten Funktionen inklusive deren Dokumentation erzeugen.

Denken Sie daran, Ihren Quellcode nach vollständiger Bearbeitung noch vor dem Beginn der Rechnerübung abzugeben. Rufen Sie hierzu in Ihrem `build`-Verzeichnis `make submit` auf.

#### 1.1 Zeitmessung mit der `libEZS`:

Um die zeitlichen Abläufe im System messen zu können, muss zunächst die `libEZS` erweitert werden. Daher beziehen sich die meisten Dateinamen in dieser Aufgabe auf Dateien im Unterverzeichnis `libEZS`. Einen hardwareunabhängigen Zähler haben wir bereits vorgegeben. Sie können auf diesen mittels der Funktion `ezs_counter_get()` zugreifen und den aktuellen Wert in Ticks auslesen. Lesen Sie die Dokumentation der von uns bereitgestellten Funktionen!

#### Aufgabe 1

Implementieren Sie die Funktionen `ezs_watch_start()` und `ezs_watch_stop()` in `libEZS/src/ezs_stopwatch.c`. Beachten Sie hierbei auch die vorgegebene Signatur und Kommentare in `stopwatch.h`. Was bedeuten die angegebenen Datentypen für Ihre zukünftigen Messungen?

Antwort:

- Zeitmessung:
  - Cyclecounter
  - Messumgebung
- WCET-Simulation
- ISR/DSR
- Antwortzeit



## AUFGABE 3: AUSFÜHRUNGSZEIT

In dieser Übungsaufgaben werden Sie sich mit verschiedenen Möglichkeiten befassen, maximale Ausführungszeiten abzuschätzen. Ziel dieser Aufgabe ist es, ein Gefühl für die Grenzen dieser Herangehensweisen zu bekommen.

### 1 Aufgabenstellung

Nach dem erfolgreichen Aufsetzen des `build`-Verzeichnisses können Sie sich mittels `make doc` eine Übersicht über alle in `libEZS` bereitgestellten Funktionen inklusive deren Dokumentation erzeugen.

Denken Sie daran, Ihren Quellcode nach vollständiger Bearbeitung noch vor dem Beginn der Rechnerübung abzugeben. Rufen Sie hierzu in Ihrem `build`-Verzeichnis `make submit` auf.

#### 1.1 Zeitmessung mit der `libEZS`:

Wir haben für Sie bereits die Funktionen `checksum()` und `bubblesort()` implementiert.

##### Aufgabe 1

Von welchen Faktoren ist die Ausführungszeit dieser Funktionen abhängig? Unterscheiden sich diese Faktoren?

Antwort:

##### Aufgabe 2

Für welche Eingaben weist `bubblesort()` die größtmögliche Laufzeit auf?

Antwort:

Verwenden Sie die von Ihnen implementierte Stoppuhr um die *Worst Case Execution Time (WCET)* beider Funktionen abzuschätzen. Implementieren Sie eine

- Zeitmessung
  - Intern: Cyclecounter
  - Extern: Oszilloskop
- Algorithmik, Eingabeabhängigkeiten
- Statistische Auswertung
- WCET
- Statische Laufzeitanalyse
- Genese von Worst-Case-Eingaben



# Aufgabe 4: Simple Scope

ÜBUNGEN ZU ECHTZEITSYSTEME

01.06.2022

## AUFGABE 4: SIMPLE SCOPE

In den vorangegangenen Übungsaufgaben haben Sie bereits periodische Aufgaben kennengelernt. Bislang erfolgte deren Implementierung durch relative Verzögerung der Fäden. In dieser Aufgabe geht es nun um die ordentliche Umsetzung eines periodischen Aufgabensystems mit den in der Übung vorgestellten Funktionen des eCos-Betriebssystems.

Im Verlauf dieser Übung werden Sie das Gerüst für ein einfaches Oszilloskop mit folgendem Funktionsumfang implementieren:

- Abtasten zweier Signale
- Berechnung der Leistungsdichtespektrums<sup>1</sup> für ein Signal
- Darstellung von Zeit- und Frequenzbereich auf einem (simulierten) Display

Die Anforderungen sollen durch das folgende System periodischer Aufgaben umgesetzt werden:

Aufgabe	Bezeichnung	Periode / ms	WCET / ms	Priorität
$T_1$	Abtastung Signal 1	10	2	-
$T_2$	Abtastung Signal 2	20	2	-
$T_3$	Analyse	20	4	-
$T_4$	Darstellung	40	6	-

Die Aufgaben verfügen über implizite Termine zu Beginn ihrer nächsten Periode. Neben dem Ihnen bereits bekannten (ereignisgesteuerten) eCos verwenden Sie in dieser Übungsaufgabe zusätzlich die zeitgesteuerte Variante `tr-eCos`. Nach dem Entpacken befinden sich diese Varianten jeweils in den Unterverordnern `event-triggered` beziehungsweise `time-triggered`. Beide Varianten unterscheiden sich hinsichtlich der API lediglich in der Umsetzung der Fäden. Den Link zur jeweiligen Funktionsreferenz finden Sie in den allgemeinen Hinweisen.

Auch in dieser Aufgabe benötigen Sie die Funktion `ezs_simulate_wcet`, sowie jeweils eine passende Umrechnung zwischen Realzeit und `(cyg, ezs, tl)_ticks`. Sie können diese mittels der bereitgestellten Tests testen. Verwenden Sie wo möglich Ihre Implementierung aus den vorangegangenen Aufgaben.

**Achten Sie bei der Bearbeitung der Übungsaufgabe darauf, dass Ihre Messungen auch bei der Abgabe noch nachvollziehbar sind!**

<sup>1</sup>Betragsquadrat der kurzzeit Fourier-Transformation, (engl. *short-time Fourier-transform, STFT*)

- Antwortzeitanalyse
- Ereignisgesteuerte Echtzeitsysteme
  - Alarme
  - RMA
  - Einplanung mit statischen Prioritäten
- Zeitgesteuerte Echtzeitsysteme
  - Ablauf tabellen
- EDF-Scheduler implementieren



# Aufgabe 5: Cyclic Scope

ÜBUNGEN ZU ECHTZEITSYSTEME

08.06.2022

## AUFGABE 5: CYCLIC SCOPE

**Hinweis:** Die *Basisaufgabe* ist eine reine Textaufgabe, eine konkrete Implementierung ist nicht erforderlich.

### 1 Aufgabenstellung

Nach dem erfolgreichen Aufsetzen des `build`-Verzeichnisses können Sie sich mittels `make doc` eine Übersicht über alle in `libE2S` bereitgestellten Funktionen inklusive deren Dokumentation erzeugen.

Denken Sie daran, Ihren Quellcode nach vollständiger Bearbeitung noch vor dem Beginn der Rechnerübung abzugeben. Rufen Sie hierzu in Ihrem `build`-Verzeichnis `make submit` auf. In dieser Übungsaufgabe sollen die unter der Bezeichnung „Zyklische Ablaufpläne“ vorgestellten Konzepte auf unser einfaches Oszilloskop angewendet werden. Grundlage bildet das aus Aufgabe 4 bekannte System von periodischen Aufgaben, nun allerdings mit größerer Last für die Darstellung:

Aufgabe	Bezeichnung	Periode ms	WCET ms	Priorität ms
$T_1$	Abtastung Signal 1	10	2	–
$T_2$	Abtastung Signal 2	20	2	–
$T_3$	Analyse	20	4	–
$T_4$	Darstellung	40	12	–

Die Aufgaben verfügen über implizite Termine zu Beginn ihrer nächsten Periode. Behalten Sie in den folgenden Teilaufgaben die Ziele der zyklischen Ablaufplanung im Hinterkopf. Bearbeiten Sie die Problemstellungen konstruktiv, halten Sie Ihr Vorgehen für die Abgabe geeignet fest (grafisch, textuell, ...).

#### 1.1 Planbarkeitsanalyse:

Zunächst ist die grundsätzliche Frage der Planbarkeit des periodischen Aufgabensystems zu klären. Vergeben Sie hierfür statische Prioritäten für die Aufgaben gemäß des aus der Vorlesung bekannten *Ratenmonotonen Algorithmus (RMA)*.

#### Aufgabe 1

Aus der Vorlesung kennen Sie die Planbarkeitsanalyse nach CPU-Auslastung. Ist das oben vorgegebene Aufgabensystem mittels RMA planbar? Führen Sie dazu die Rechenzeitbedarfsanalyse durch.

- Planbarkeitsanalyse durch CPU-Auslastung
- Zyklische Ablaufpläne
  - ↳ Rahmenmodell
- Nicht-periodische Aufgaben
  - Unterbrecherbetrieb
  - Hintergrundbetrieb
  - Periodischer Zusteller
  - Slack-Stealing
- Implementierung Cyclic Executive



# Aufgabe 6: Extended Scope

ÜBUNGEN ZU ECHTZEITSYSTEME

29.06.2022

## AUFGABE 6: EXTENDED SCOPE

In dieser Aufgabe wird das Oszilloskop um die Ausgabe des Zeitbereichsignals und eine Befehlschnittstelle erweitert. Hierfür wird eCos im ereignisgesteuerten Betrieb verwendet – vergeben Sie die Prioritäten nach dem Rate Monotonic Algorithm (RMA).

Bezeichnung	Periode ms	WCET ms
$T_1$ Abtastung Signal	4	0,5
$T_2$ Flankenerkennung	10	1,0
$T_3$ Analyse PDS	1000	?
$T_4$ Darstellung Signal	250	?
$T_5$ Darstellung PDS	1000	?

Die Deadline entspricht jeweils der Periode.

$T_1$  Tastet den ADC ab (`ezs_adc_get()`) und fügt die Werte in einen Ringpuffer der Größe 64 (`TIME_DOMAIN_LENGTH`) ein.

$T_2$  Simuliert in der einfachen Übung nur Laufzeit.

$T_3$  Liest die Werte aus dem Ringpuffer und ruft zur PDS-Analyse die vorgegebene Funktion `ezs_easy_pds()` auf.

$T_4$  Stellt das abgetastete Signal mithilfe der vorgegebenen Funktion `ezs_plot()` auf dem Framebuffer dar.

$T_5$  Stellt die Ergebnis der PDS-Analyse durch Aufruf der ebenfalls gegebenen Funktionen `ezs_plot_pds()` auf dem Framebuffer dar.

Für den Datenaustausch zwischen  $T_1$ ,  $T_3$ ,  $T_4$  und  $T_5$  bieten sich globale Arrays an. Die Funktion `ezs_easy_pds()` legt ihre Ausgangswerte in ein Array der Größe 32 ab. Dieses können Sie direkt in `ezs_plot_pds()` nutzen. Die genaue Schnittstellenbeschreibung können Sie den Headerdateien entnehmen. Sie finden das oben genannte Aufgabensystem bereits in weiten Teilen in der Vorgabe, auch einige grundlegende globale Puffer und Daten sind von uns bereits angelegt worden. Die Vorgabe soll Ihnen die Aufgabe erleichtern, Sie müssen sich aber nicht zwingend daran halten. Globale Definitionen finden sich zur besseren Lesbarkeit in der Datei `config.h`, welche Sie ebenfalls im Projektverzeichnis vorfinden.

- Aperiodische Steuerung
- Logische Ereignisse/Aktivierungen
- Minimale Zwischenankunftszeiten
- Rangfolge
- Betriebs(modus-)wechsel
- Zustandsmaschinen



# Aufgabe 7: Zugriffskontrolle

ÜBUNGEN ZU ECHTZEITSYSTEME

06.07.2022

## AUFGABE 7: ZUGRIFFSKONTROLLE

In dieser letzten Übungsaufgabe werden Sie sich mit gegenseitigem Ausschluss und Zugriffskontrolle befassen. Diese Übungsaufgabe zielt auf die Probleme der Prioritätsumkehr und der Verklemmung ab und dient dazu die in der Vorlesung vorgestellten echtzeitfähigen Synchronisationsprotokolle (vgl. VII 11 ff.) praktisch anzuwenden.

In dieser Übung greifen wir der Einfachheit halber auf die folgenden synthetischen Aufgabensysteme zurück:

Tabelle 1: Aufgabensystem 1 – „Pathfinder“

Aufgabe	Periods ms	Phase ms	WCET ms	Betriebsmittel <sup>1</sup>
T <sub>1</sub>	20	4	6	(R <sub>1</sub> , 3, 1)
T <sub>2</sub>	50	3	4	
T <sub>3</sub>	200	1	9	(R <sub>1</sub> , 1, 7)

Tabelle 2: Aufgabensystem 2

Aufgabe	Periode ms	Phase ms	WCET ms	Betriebsmittel <sup>1</sup>
T <sub>4</sub>	20	7	2	(R <sub>2</sub> , 1, 1)
T <sub>5</sub>	50	5	6	(R <sub>2</sub> , 1, 5) (R <sub>3</sub> , 3, 2)
T <sub>6</sub>	100	3	6	(R <sub>3</sub> , 1, 5) (R <sub>4</sub> , 4, 2)
T <sub>7</sub>	200	1	10	(R <sub>4</sub> , 1, 9)

<sup>1</sup>Notation Betriebsmittel: (Betriebsmittel, relativer Anforderungszeitpunkt, Haltezeit)

- Betriebsmittel und kritische Abschnitte
- Verklemmungen
- Prioritätsumkehr
- Blockadezeit
- Synchronisationsprotokolle
  - Verdrängungssteuerung
  - Prioritätsvererbung
  - Prioritätsobergrenzen



- 1 Werbung
- 2 Rekapitulation
- 3 Fragen zur Klausur**
- 4 Klausurvorbereitung



- 1 Werbung
- 2 Rekapitulation
- 3 Fragen zur Klausur
- 4 Klausurvorbereitung**



jeopardy.cs.fau.de

