## AUFGABE 3: ANTWORTZEIT

Diese Aufgabe soll Ihnen ein erstes Gefühl für das Zusammenspiel von periodischen und nichtperiodischen Ereignissen in einem Echtzeitsystem vermitteln.

### *Aufgabenstellung*

**1**. *WCET-Simulation:* Um zukünftig den Rechenzeitaufwand einer komplexeren Anwendung zu simulieren, ist ein weiteres Instrument notwendig. Implementieren Sie hierfür die Funktion `ezs_lose_time()`, welche aktiv wartet und erst nach dem Verbrauchen einer bestimmten CPU-Zeit zurückkehrt. Setzen sie zur Zeitkontrolle wiederum den bereitgestellten Zähler ein und implementieren Sie den in der Tafelübung präsentierten Algorithmus – ihre Funktion muss mit Unterbrechungen umgehen können. Testen Sie, ob Ihre `ezs_lose_time()` die Zeitvorgabe einhält! Erweitern Sie `ezs_lose_time()` nun so, dass ein zufällig gewählter Anteil der gewählten Zeit *nicht* verbraucht wird. Die maximale Grösse dieses zufälligen Anteils soll über der zweiten Parameter der Funktion wählbar sein.

☞ ezs_stopwatch.c

**2**. *Signalerzeugung:* Die Rekonstruktion von kontinuierlichen Signalen, beispielsweise von Musikstücken, ist eine typische Aufgabe eines Echtzeitsystems. Hierbei gilt das Nyquist-Shannonsche Abtasttheorem[1], wonach für eine korrekte Rekonstruktion des zeitdiskreten Signals die Abtastfrequenz $f_{\text{sample}}$ so zu wählen ist, dass sie mindestens $2 \cdot f_{\text{max}}$ (Maximalfrequenz des Signals) ist. Bei der Wiedergabe digitalisierter Musikstücke ist diese Abtastfrequenz (sampling rate) von entscheidender Bedeutung für die verzerrungsfreie Rekonstruktion analoger Signale. Der Einhaltung des Abtasttheorems fällt also eine zentrale Bedeutung bei der Implementierung eines Echtzeitsystems zu. Verwenden Sie, ähnlich wie in der vorangegangenen Aufgabe, die Funktion `sinf()` um ein überlagertes Sinus-Signal zu erzeugen. Dieses soll eine Komponente mit einer Frequenz von 2 Hz und eine mit 13 Hz enthalten. Geben Sie den errechneten Signalverlauf auf dem Analog-Digital-Umsetzer so aus, dass das erwünschte analoge Signal korrekt rekonstruiert wird. *Mit welcher Rate müssen Sie die Abtastwerte wiedergeben? Was passiert, wenn Sie diese Rate verdoppeln? Was passiert, wenn Sie die Rate mit einer gleichverteilten zufälligen Abweichung von bis zu 90% nicht einhalten (Betrachten Sie auch am Oszilloskop die Fouriertransformierte des Signals)? Verwenden Sie hierfür* `ezs_lose_time()` *um zusätzliche Rechenlast zu simulieren.*

**3**. *Antwortzeit:* Unter Antwortzeit versteht man die Zeit zwischen dem Auftreten eines Ereignisses (z. B. eines Interrupts) und dem Bereitstellen eines Ergebnisses (z. B. Öffnen eines Ventils) durch das Echtzeitsystem. Die Antwortzeit hängt dabei

[1] `http://de.wikipedia.org/wiki/Abtasttheorem`

nicht alleine vom Rechenzeitbedarf der Ereignisbehandlung sondern von vielen Faktoren ab.
In dieser Aufgabe setzen wir für die Erzeugung von Ereignissen die serielle Schnittstelle ein. Diese teilt durch das Auslösen eines Interrupts mit, wenn ein Zeichen eingelesen wurde. Wir haben den entsprechenden Interrupt bereits in der Vorgabe für Sie aufgesetzt und eine rudimentäre Interruptbehandlung implementiert. *Inwiefern ist die von uns bereitgestellte Interruptbehandlung für ein komplexes Echtzeitsystem nicht geeignet?* Implementieren Sie nun eine bessere Interruptbehandlung, die die eingelesenen Zeichen mit Hilfe der Funktion `diag_printf()` auf der seriellen Schnittstelle ausgibt.
Vergeben Sie die Prioritäten für diese neue Aufgaben so, dass die Signalerzeugung aus der vorangegangenen Teilaufgabe eine niedrigere Priorität (größere Zahl) erhält. Verwenden Sie die Zeitmessung per Systemzeitgeber um die Antwortzeit zwischen dem Auftreten des Interrupts und der Ausgabe zu ermitteln. Nutzen Sie hierbei die cutecom-Funktion „Send File“ um die serielle Schnittstelle auszulasten. *Speichern Sie die in beiden Fällen gemessenen Werte nach der Messung über die serielle Schnittstelle auf Ihrem Desktop-PC ab. Was beobachten Sie?*
Als letztes soll die Abtast-Aufgabe eine höhere Priorität (kleinere Zahl) als das Auslesen der seriellen Schnittstelle erhalten. Darüberhinaus soll sie zusätzliche Rechenzeit benötigen. Verwenden sie hierfür die Funktion `ezs_lose_time()` und wählen sie als Zeit $\frac{1}{5}$ der Periode der Abtastaufgabe (z. B. ergibt eine Periode von 1 ms → eine WCET von 200 µs). *Wie verhält es sich jetzt mit der Antwortzeit? Wodurch kommt der Unterschied zur vorherigen Messung zustande?*

*Hinweise*

- Bearbeitung: Gruppe mit je zwei/drei Teilnehmern.
- Abgabezeit: 17.11.2014
- Fragen bitte an `i4ezs@lists.cs.fau.de`