AUFGABE 2: ANTWORTZEIT

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

Aufgabenstellung

1. *Zeitmessung mit der libEZS:* Um die zeitlichen Abläufe im System messen zu können, muss zunächst die `libEZS` erweitert werden. 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. *Implementieren Sie nun die Funktionen* `ezs_watch_start()` *und* `ezs_watch_stop()` *in* `ezs_stopwatch.c`. In der schon erwähnten Dokumentation werde auch die Signaturen dieser Funktionen beschrieben. *Was bedeuten die angegebenen Datentypen für Ihre zukünftigen Messungen?* Damit mehrere Zeitmessungen parallel und über Funktionsgrenzen hinweg erfolgen können, erhalten beide Funktionen einen Zeiger auf den Zustand der Messung durch den Aufrufer. Die Zustandsvariable, die diesen Zustand abbildet, muss später in der Anwendung global[1] angelegt werden. Die Funktion `ezs_watch_stop()` liefert das Ergebnis der Messung in Form von Zeitgeber-Ticks zurück. Die Dauer eines solchen Ticks ist prinzipiell hardwareabhängig, die Auflösung unseres Zählers gibt Ihnen die Funktion `ezs_counter_resolution_ps()` zurück.

☞ `libEZS/...`

☞ `ezs_counter.h`

☞ `make doc`

2. *WCET-Simulation:* Um in dieser und den folgenden Aufgaben möglichst reale Anwendungssysteme betrachten zu können, müssen wir den Rechenzeitaufwand komplexer Anwendung simulieren können. 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. Stellen Sie sicher, dass ihre Funktion nie *mehr* Zeit verbraucht als der übergebene Zeitwert vorschreibt. *Weshalb ist dies sinnvoll?* Testen Sie mit der in der vorherigen Aufgabe entwickelten Funktion, ob Ihre `ezs_lose_time()` die Zeitvorgabe einhält!

☞ `ezs_stopwatch.c`

[1] Gültigkeit von Variablen in C, Folie 37ff: `https://www4.cs.fau.de/Lehre/SS13/V_SP1/Folien/SP-02a-A4.pdf`

Erweitern Sie `ezs_lose_time()` nun so, dass ein zufällig gewählter Anteil der angefragten Zeit *nicht* verbraucht wird. Die maximale Grösse dieses zufälligen Anteils soll über der zweiten Parameter der Funktion wählbar sein.

3. *Signalerzeugung:* Die Rekonstruktion (Wiedergeben) von kontinuierlichen Signalen, beispielsweise von Musikstücken, ist eine typische Aufgabe eines Echtzeitsystems. Hierbei gilt das Nyquist-Shannonsche Abtasttheorem[2], wonach für eine korrekte Rekonstruktion eines zeitdiskreten Signals die Abtastfrequenz f_{sample} so zu wählen ist, dass sie größer $2 \cdot f_{\text{max}}$ (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 minimalen Rate müssen Sie die Abtastwerte wiedergeben? Was passiert, wenn Sie diese Rate verdoppeln? Verwenden Sie Ihre* `ezs_lose_time()` *um zusätzlich zu Ihrer mit* `cyg_thread_delay()` *eingebrachten Verzögerung Rechenlast zu simulieren. Was passiert, wenn durch diese eingebrachte Rechenlast die Rate mit einer gleichverteilten zufälligen Abweichung von bis zu 90% nicht mehr eingehalten wird? Betrachten Sie hierbei auch die Fouriertransformierte des Signals am Oszilloskop.*

4. *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 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 unter Nutzung der in der Übung vorgestellten Konzepte *DSR* und *Faden* eine bessere Interruptbehandlung, die die eingelesenen Zeichen mit Hilfe der Funktion `diag_printf()` auf der seriellen Schnittstelle im Faden-Kontext ausgibt.

Vergeben Sie die Prioritäten für diese neue Aufgaben so, dass die Signalerzeugung aus der vorangegangenen Teilaufgabe eine niedrigere Priorität (größere Zahl) erhält.

[2] `https://de.wikipedia.org/wiki/Abtasttheorem`

Verwenden Sie die zuvor implementierte Zeitmessung per Systemzeitgeber, um die Antwortzeit zwischen dem Auftreten des Interrupts und der Ausgabe zu ermitteln. *Wo beginnt die Messung? Wo ist sie zu Ende?* 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. *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: 18.11.2015
- Fragen bitte an `i4ezs@lists.cs.fau.de`