AUFGABE 3: SIMPLE SCOPE

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

Grundlage dieser Übungsaufgabe ist ein einfaches Oszilloskop welches in der Lage ist Signale abzutasten. Darüber hinaus soll es auch eine einfache Analyse ermöglichen und eines der Signale mittels einer diskreten Fourier-Transformation vom Zeit- in den Frequenzbereich zu transformieren. Weiterhin soll das Ergebnis dieser Analyse auf einem Display ausgegeben werden.

Für die Umsetzung soll in einem ersten Schnitt zunächst das hierfür notwendige System von periodischen Aufgaben implementiert werden. Gegeben sei:

Aufgabe	**Bezeichnung**	**Periode /** ms	**WCET /** ms
T_1	Abtastung Signal 1	10	2
T_2	Abtastung Signal 2	20	2
T_3	Analyse	20	3
T_4	Darstellung	100	6

Ab dieser Übungsaufgabe finden Sie in der Vorgabe neben dem Ihnen bereits bekannten eCos zusätzlich noch die zeitgesteuerte Variante tt-eCos. Sie finden beide nach dem entpacken unter `event-triggered` beziehungsweise `time-triggered`. Die Varianten unterscheiden sich hinsichtlich der API lediglich in der Umsetzung der Fäden. Den Link zur jeweiligen Funktionsreferenz finden Sie in den allgemeinen Hinweisen.

Aufgabenstellung

A. *Implementierung:* Implementieren Sie das gegebene Aufgabensystem und simulieren Sie dabei die angegebene Laufzeit jedes Fadens mithilfe von `ezs_watch_wcet()` (die Aufgaben tun sonst

nichts weiter!). Um der Natur einer **Worst-Case**-Execution-Time näher zu kommen, soll der Wert in Zukunft um bis zu 30% schwanken. Erweitern Sie daher die Funktion `ezs_watch_wcet()` so, dass sie das gewünschte Verhalten aufweist. Verwenden Sie die Funktion `rand()` um eine zufälligen Wert von der WCET **abzuziehen**. Der von `rand()` ausgegebene Wertebereich liegt zwischen `0` und `RAND_MAX` (siehe Manpage: `man 3 rand`). Sie müssen die Zufallszahl daher noch auf 0 bis 30% der WCET skalieren (z. B. durch eine Modulo-Operation).

☞ `stdlib.h`

B. *Ereignisgesteuerte Umsetzung (eCos):* Integrieren Sie die Aufgaben in die Ihnen bereits bekannten, **ereignisgesteuerten** Variante von eCos. Vergeben Sie die Prioritäten gemäß des in der Vorlesung vorgestellten **Rate-Monotonic-Algorithmus**. Verwenden Sie für die periodische Aktivierung der Fäden die von eCos bereitgestellten Alarme (siehe Folien zur Übung / API), initialisieren Sie diese zunächst mit einem `trigger` von `cyg_current_time()` + 0.
Verwenden Sie nun die in Aufgabe 2 implementierte Zeitmessung oder das nun zur Verfügung stehende Software-Tracing um die Periode von T_2 und T_3 zu messen. Zeichnen Sie diese für mindestens 5 Zyklen so auf (aufschreiben, Textdatei, Screenshot, ...), dass Ihre Messung bei der Abgabe nachvollziehbar ist. *Was beobachten Sie?*
Aus den gewonnenen Daten lässt sich ein verbesserter Ablaufplan, wie er in der Vorlesung und in der Übung vorgestellt wurde, erstellen, der die Beeinflussung der einzelnen Fäden untereinander vermindert oder sogar unterbindet. Ändern Sie die Alarme entsprechend ab und wiederholen Sie die Messung.
Variieren Sie den `trigger`*-Wert. Welches Vorgehen haben Sie gewählt und welches Vorwissen war dafür nötig? Was beobachten Sie?*

☞ `.../event-triggered`

☞ `ezs_watch_...()`

☞ 100 ms

☞ getrennte Aufzeichnung!

C. *Taktgesteuerte Umsetzung (tt-eCos):* Im nächsten Schritt soll das Aufgabensystem in die zeitgesteuerte Variante tt-eCos integriert werden. *Stellen Sie einen geeigneten Ablaufplan auf und nehmen Sie dabei an, dass* ***Deadline = Periode - 1****. Sorgen Sie von Anfang an dafür, dass sich die Fäden nicht überlappen;* nutzen Sie hierfür ihr bereits gewonnenes Wissen aus der vorangegangenen Teilaufgabe. *Führen Sie wiederum eine Messung der Periode von* T_2 *und* T_3 *durch und zeichnen Sie diese auf. Was beobachten Sie? Welche Vorbeziehungsweise Nachteile sehen sie zur ereignisgesteuerten Lösung?*

☞ `.../time-triggered`

Beachten Sie, dass Fäden in tt-eCos eine run-to-completion Semantik aufweisen, d. h. sie beenden sich nach jedem Durchlauf und werden vom Scheduler neu gestartet. Auch in tt-eCos haben wir die interne Uhr auf 1 ms eingestellt → 1 `tt_ticktype` = 1 ms.

- Die Implementierung der Ablauftabelle in tt-eCos erlaubt nicht das Auslösen mehrerer Fäden/Deadlineüberpruefungen zu einem Zeitpunkt.
- Achten Sie auf eine ausreichend groß angelegte Tabelle ⇒ `tt_DispatcherTable(table1, XXX);`
- Die Deadlineüberprüfung kann auch schon vor dem ersten Start geplant werden, sie wird dann erst in der nächsten Hyperperiode aktiv.

Erweiterte Aufgabe

A. *EZS-Simple-Scope:* In den erweiterten Übungen soll nun das *EZS-Simple-Scope* konkret implementiert werden. Um das in der Übung vorgestellten Leistungsdichtespektrums (LDS) korrekt zu implementieren, müssen Sie $N = 2^{\mu}$ Werte abtasten bevor Sie das LDS berechnen können. Hierfür sollen Sie von dem folgende, angepassten System von Aufgaben ausgehen ($N = 64$):

Aufgabe	**Bezeichnung**	**Periode /** ms	**WCET /** ms
T_1	Abtastung Signal 1	16	2
T_2	Abtastung Signal 2	4	?
T_3	Analyse	256	?
T_4	Darstellung	256	?

Die Implementierung soll in der ereignisgesteuerten Variante erfolgen. Sichern Sie zu diesem Zweck Ihre Implementierung aus den vorangegangenen Teilaufgaben für die spätere Abgabe. Passen Sie die Prioritäten gemäß des RMA an und lassen sie die Aufgaben wieder zum Zeitpunkt `cyg_current_time()` + 0 starten. Weiterhin soll nun T_2 den ADC auslesen und die Werte in das Eingangsarray des LDS schreiben (T_1 **verbleibt als Dummy-Faden**). Konvertieren Sie die Eingangswerte des ADC (0 bis 255) in einen Wertebereich von -0,5 bis 0,5 (Achtung bei der Umwandlung von `float` ↔ `integer`).

Ersetzen Sie nun die Implementierung von T_3 durch die in der `libEZS` bereitgestellte LDS-Funktion. Die Ausgabe hat näherungsweise einen Wertebereich ist von $-140\,\mathrm{dB}$ bis $0\,\mathrm{dB}$. Mit dem o. g. Aufgabensystem ($N = 64$) können Sie näherungsweise davon ausgehen, dass die Anzeige (alle 0,25 s) mit einer Auflösung von 1 Hz im LDS erfolgt.
Das Ein- und Ausgabearray des LDS muss also 64 Elemente groß sein. Sie können allerdings nur die ersten 32 Werte der Ausgabe verwenden. Überprüfen Sie ihre Implementierung (`printf()`, Debugger, Tracer, ...).

B. *Ausgabe:* Geben Sie das Ergebnis der Analyse nun auf dem grafischen Display (Framebuffer) aus, skalieren Sie die Ausgabe der Werte (dB) entsprechend der Displaygröße (Pixel). Implementieren Sie die Anzeige in T_4 gemäß dem in der Übung vorgeschlagenen Aufbau – die konkrete Ausgestaltung der Anzeige ist Ihnen überlassen. In unserem Beispiel erhalten Sie 32 Werte, wobei die Frequenz in ~0,5 Hz Schritte unterteilt ist. Die Werte im Spektrum entsprechen dem Leistungspegel und werden in der Pseudo-Einheit Dezibel (dB) angegeben.

C. *Analyse:* Im letzten Schritt dieser Aufgabe soll nun abermals die Überlappung der Aufgaben unterbunden werden. Wir gehen hierfür davon aus, dass nur noch T_1 künstlich durch eine WCET verlängert wird. Die Ausführungszeit der anderen Aufgaben muss von Ihnen abgeschätzt werden (z. B. mittels des Tracers). Erstellen Sie mithilfe Ihrer Messungen einen neuen Ablaufplan und passen Sie den Phasenversatz entsprechend an. Wo sehen Sie in der Praxis die Probleme? Was würde hinsichtlich der Analyse und Planung passieren, wenn Sie die Perioden nicht harmonisch wählen (z.B. tatsächlich 0,25 s für die Ausgabe)?

Wichtig:

- In dieser Aufgabe entspricht ein Tick in eCos wie in tt-eCos 1 ms.
- Fäden unterscheiden sich zwischen eCos (Schleife) und tt-eCos (run-to-completion).

- Die Auflösung des libEZS-Zählers (ezs_counter_get()) beträgt 1 µs.