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.

In dieser Übungsaufgabe werden Sie ein einfaches Oszilloskop implementieren, 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 (DFT) vom Zeit- in den Frequenzbereich transformieren. Weiterhin soll das Ergebnis dieser Analyse auf einem Display ausgegeben werden.

Für die Umsetzung soll in einem ersten Schritt 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

In 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.

Auch in dieser Aufgabe benötigen Sie wieder die Funktion `ezs_lose_time`. Verwenden Sie Ihre Implementierung aus den vorangegangenen Aufgaben. **Achten Sie bei der Bearbeitung dieser Übungsaufgabe darauf, dass Ihre Messungen auch bei der Abgabe noch nachvollziehbar sind!**

1 Aufgabenstellung

Implementieren Sie das gegebene Aufgabensystem und simulieren Sie dabei die angegebene Laufzeit jedes Fadens mit Hilfe von `ezs_lose_time()`. Verwenden Sie `ezs_lose_time()` so, dass die Ausführungszeiten um bis zu 100% schwanken. Beachten Sie, dass wir in den grundlegenden Übungen nur die Aufgaben-*Struktur* betrachten, die Aufgaben selbst jedoch nicht implementiert werden sollen.

1.1 Ereignisgesteuerte Umsetzung (eCos):

Integrieren Sie die Aufgaben in die Ihnen bereits bekannte, *ereignisgesteuerte* Variante von eCos. Vergeben Sie die Prioritäten gemäß des in der Vorlesung vorgestellten *Rate-Monotonic-*

☞ .../event-triggered

Algorithmus. Verwenden Sie für die periodische Aktivierung der Fäden die von eCos bereitgestellten *Alarme*, initialisieren Sie diese zunächst mit einem `trigger` von `cyg_current_time()` + 1.

Verwenden Sie das nun zur Verfügung stehende *Software-Tracing* um die Periode von T_2 und T_3 zu messen (*Beachten Sie die Verwendungshinweise zum Tracer*). Sichern Sie die Tracing-Datei für die unterschiedlichen Ablaufpläne dieser Aufgabe für Ihre Abgabe.

☞ `make trace`

1. Aufgabe Was beobachten Sie hinsichtlich der zeitlichen Parameter der einzelnen Aufgaben? Welche Auswirkungen sehen Sie hinsichtlich der Abtastung von Signal 2?

Antwort:

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.

2. Aufgabe Ändern Sie die Alarme entsprechend ab und wiederholen Sie die Messung – variieren Sie hierzu den `trigger`-Wert. Welches Vorgehen haben Sie gewählt und welches Vorwissen war dafür nötig? Was beobachten Sie?

☞ getrennte Aufzeichnung!

Antwort:

1.2 Taktgesteuerte Umsetzung (tt-eCos):

Im nächsten Schritt soll das Aufgabensystem in die zeitgesteuerte Variante tt-eCos integriert werden.

☞ `.../time-triggered`

Lesen Sie die folgenden Hinweise bevor Sie mit der Bearbeitung der zeitgesteuerten Umsetzung anfangen:

- Es ist leider nicht möglich, zwei Aufgaben zum selben Zeitpunkt einzuplanen. Dies betrifft auch Deadlineüberprüfungen. Sie dürfen deswegen davon ausgehen, dass $D_i = p_i - 1\,\text{ms}$.
- Achten Sie auf eine ausreichend groß angelegte Tabelle

  ⇒ `tt_DispatcherTable(table1, XXX);`

  Für jede Ereignisbehandlung (d. h. die eigentlich Behandlung *und* die Deadlineüberprüfung) muss Platz in der Ablauftabelle sein.
- Die Deadlineüberprüfung kann auch schon *vor* dem ersten Start geplant werden, sie wird dann erst in der nächsten Hyperperiode aktiv.

- Der Tracer stellt Prioritätsebenen dar und erlaubt somit keine Unterscheidung von Aufgaben auf einer Prioritätsebene. Weiten Sie ggf. den Prioritätenraum des eingesetzten Planungsalgorithmus auf um Aufgaben unterscheidbar zu machen.
- Das Tracing zeigt Ihnen die ersten 256 Einlastungsereignisse des Systems an. Die Daten liegen in der Datei `tracetmp/tracefile` in Ihrem `build`-Ordner.

Stellen Sie einen geeigneten Ablaufplan auf und planen Sie für jede Aufgabe eine Deadlineüberprüfung ein! Sorgen Sie dabei von Anfang an dafür, dass sich die Fäden nicht überlappen; nutzen Sie Ihr bereits gewonnenes Wissen aus der vorangegangenen Teilaufgabe.

3. Aufgabe Führen Sie wiederum eine Messung der Periode von T_2 und T_3 durch und zeichnen Sie diese auf. Was beobachten Sie?

Antwort:

4. Aufgabe Welche Vor- beziehungsweise Nachteile sehen sie gegenüber der ereignisgesteuerten Lösung?

Antwort:

2 Erweiterte Aufgabe

2.1 EZS-Simple-Scope:

In den erweiterten Übungen soll nun das *EZS-Simple-Scope* konkret implementiert werden. Um das in der Übung vorgestellte Leistungsdichtespektrums (LDS) korrekt zu implementieren, müssen Sie N-Werte abtasten bevor Sie das LDS berechnen können. Da der Algorithmus nur Zweierpotenzen als Eingabelängen akzeptiert, sollen Sie von $N = 64$ ausgehen, woraus sich das folgende, angepasste System von Aufgaben ergibt:

exTask	**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()` + 1 starten.

5. Aufgabe 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 bisherige Implementierung von T_3 durch die in der `libEZS` bereitgestellte LDS-Funktion. Die Ausgabe hat näherungsweise einen Wertebereich von −140 dB bis 0 dB. Mit dem oben genannten Aufgabensystem ($N = 64$) können Sie näherungsweise davon ausgehen, dass die Anzeige (alle 0,25 s) mit einer Auflösung von 1 Hz im LDS erfolgt.

☞ `ezs_adc_get()`

☞ `ezs_power_density_spect`

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, ...). **Achten Sie bei der Verwendung des LDS darauf, dass die Eingabedaten, die der LDS-Funktion übergeben werden in ihrer korrekten zeitlichen Reihenfolge sortiert sind!**

6. Aufgabe 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 des in der Übung vorgeschlagenen Aufbaus – 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.

☞ `ezs_fb_*()`

2.2 Analyse:

Im letzten Schritt dieser Aufgabe soll nun abermals die Überlappung der Aufgaben nach Möglichkeit 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 gemessen werden.

7. Aufgabe Erstellen Sie mit Hilfe Ihrer Messungen einen neuen Ablaufplan und passen Sie den Phasenversatz entsprechend an.

8. Aufgabe Was würde hinsichtlich des Aufwands für Analyse und Planung passieren, wenn Sie z. B. 257 ms als Periode für die Ausgabe und 7 ms für die Abtastung wählen?

Antwort:

9. Aufgabe Wie sollten die Perioden von Aufgaben also in der Praxis ausgewählt werden?

Antwort:

Hinweise

- Bearbeitung: Gruppe mit je drei Teilnehmern.
- Abgabefrist: 15.12.2017
- Fragen bitte an `i4ezs@lists.cs.fau.de`