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 ein einfaches Oszilloskop mit folgendem Funktionsumfang implementieren:

- Abtasten zweier Signale
- Berechnung der Leistungsdichtespektrums[1] 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
T_1	Abtastung Signal 1	10	2
T_2	Abtastung Signal 2	20	2
T_3	Analyse	20	3
T_4	Darstellung	100	6

Neben dem Ihnen bereits bekannten (ereignisgesteuerten) eCos verwenden Sie in dieser Übungsaufgabe zusätzlich die zeitgesteuerte Variante tt-eCos. Nach dem Entpacken befinden sich diese Varianten jeweils in den Unterordnern `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 wieder die Funktion `ezs_lose_time`. Verwenden Sie Ihre Implementierung aus den vorangegangenen Aufgaben.

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

1 Aufgabenstellung

In der grundlegenden Übung werden Sie zunächst nur die Aufgaben-*Struktur* betrachten. Implementieren Sie hierfür das gegebene Aufgabensystem und simulieren Sie die angegebenen

[1] Betragsquadrat der kurzzeit Fourier-Transformation, (*engl. short-time Fourier-transform, STFT*)

Ausführungszeiten der Fäden mit Hilfe von `ezs_lose_time()` so, dass diese um bis zu 100% schwanken. Eine Implementierung der tatsächlichen Aufgabenfunktionalität ist in diesem Schritt nicht erforderlich.

Mit dem bereitgestellten *Software-Tracing* können Sie das zeitliche Verhalten des Systems und speziell die Perioden von T_2 und T_3 bestimmen. Sichern Sie alle relevanten Tracing-Dateien für die Abgabe. Die der letzten Ausführung befindet sich unter `build/tracetmp/tracefile`.

☞ `make trace`

Hinweis: Der Tracer stellt Prioritätsebenen dar und erlaubt somit keine Unterscheidung von Aufgaben auf einer Prioritätsebene. Weiten Sie bei Bedarf den Prioritätenraum der eingesetzten Planungsalgorithmen auf um Aufgaben unterscheidbar zu machen.

1.1 Ereignisgesteuerte Umsetzung (eCos):

In der ereignisgesteuerten Umsetzung sollen die Aufgaben nach dem in der Vorlesung vorgestellten *Rate-Monotonic-Algorithmus* eingeplant werden. Verwenden Sie für die periodische Aktivierung der Fäden die von eCos bereitgestellten *Alarme*. Der Parameter `trigger` der Funktion `cyg_alarm_initialize()` soll hierbei zunächst auf `cyg_current_time() + 1` gesetzt werden.

☞ `.../event-triggered`

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 vorgestellt wurde, bestimmen, der die Beeinflussung der einzelnen Fäden untereinander vermindert oder sogar unterbindet.

2. Aufgabe Ändern Sie die Alarme entsprechend hinsichtlich des Wertes von `trigger` ab und wiederholen Sie die Messung. 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

☞ `.../time-triggered`

werden.
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.
- 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 von Anfang an dafür, dass sich die Fäden nicht überlappen.

3. Aufgabe Führen Sie wiederum eine Messung der Perioden 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 der erweiterten Übung soll das *EZS-Simple-Scope* nun konkret implementiert werden. Das in der Übung vorgestellte Leistungsdichtespektrum (LDS) benötigt N Werte, welche zuvor

abgetastet werden müssen. Der Algorithmus akzeptiert nur Zweierpotenzen als Eingabelängen. Arbeiten Sie mit $N = 64$. Die Änderungen spiegeln sich folgendermaßen im Aufgabensystem wieder:

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 T_2 soll nun 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 die bisherige Implementierung von T_3 durch die in der `libEZS` bereitgestellte LDS-Funktion. Die Ausgabe hat näherungsweise einen Wertebereich von −140 dBFS bis 0 dBFS. Mit dem oben genannten Aufgabensystem und $N = 64$ können Sie näherungsweise davon ausgehen, dass die Anzeige (alle 0,25 s) mit einer Wiederholrate von 1 Hz im LDS erfolgt.
Das Ein- und Ausgabearray des LDS muss N Elemente groß sein, wobei das Ergebnis nur die ersten $N/2$ Werte umfasst. Die übergebenen Daten müssen entsprechend ihrer zeitlichen Reihenfolge sortiert sein.

☞ `ezs_adc_get()`

☞ `ezs_power_density_spectrum()()`

6. Aufgabe Geben Sie jetzt das Ergebnis der Analyse auf dem grafischen Display (Framebuffer) aus. Skalieren Sie die Ausgabe der Werte (dBFS) 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 Dezibel relativ zur Vollaussteuerung (dBFS, *engl. Decibels relative to full scale*) angegeben.

☞ `ezs_fb_*()`
☞ `ezs_io_fel.h`

2.2 Analyse:

Im letzten Schritt dieser Aufgabe soll nun abermals die Überlappung der Aufgaben minimiert werden. Während die simulierte Ausführungszeit von T_1 durch die angegebene WCET be-

schränkt ist, *müssen die Ausführungszeiten der anderen Aufgaben 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: 6.12.2018
- Fragen bitte an `i4ezs@lists.cs.fau.de`