AUFGABENBLOCK 3 28. April 2004

BrickOS als Real-Time Multitasking-Betriebsystem:

1. BrickOS verwendet ein Round Robin Scheduling Theme. Welche Probleme können unter Berücksichtigung des Echtzeit-Aspekts mit diesem Scheduling Theme auftreten [3]?

2. Erklären Sie wie Deadlocks und Priority Inversion entstehen können und warum diese Phänomene in einer Echtzeit-Umgebung vermieden werden müssen [4] [2].

3. Wie kann man diese Probleme in den Griff bekommen [4] [2]?

4. Worin besteht der prinzipielle Unterschied zwischen den Modellen *synchronous*, *scheduled*, *timed*? Erklären Sie die Vor- und Nachteile der einzelnen Modelle [2].

5. Die E-Machine bildet die Grundlage zur Ausführung von plattformunabhängigen Echtzeitprogrammen wie TDL-Programme [1]. Implementieren Sie eine E-Machine Version für BrickOS, welche die E-Code-Instruktionen `call`, `schedule` und `future` beherrscht. Der E-Code wird in Form eines Arrays fix in den Code eingebunden. Beschreiben Sie in Ihrem Vortrag genau, wie unter Ausnutzung der Betriebssystemeigenschaften von BrickOS die E-Machine implementiert werden kann (Zeitverhalten, Scheduling, Lesen von Sensoren und Schreiben auf Aktuatoren). Wo treten Probleme auf? Implementieren Sie Ihre E-Machine so, dass im Falle einer Überschreitung von Deadlines Laufzeitfehler auftreten und entsprechende Meldungen am LCD-Display angezeigt werden. Testen Sie Ihre E-Machine mit verschiedenen E-Code-Sequenzen.

Die Fragen 1-4 sind theoretischer Natur und werden mündlich im PS überprüft. Bitte bereiten Sie zu Frage 5 wieder ein paar Präsentationsfolien vor und geben Sie das Programm in Form von Source-Code und Binaries ab. Es ist Ihnen selbstverständlich freigestellt (es ist aber kein Muß), bereits richtige Kontroll-Anwendungen mit E-Code zu erstellen. Der nächste Aufgabenblock wird sich u.a. mit dem Entwurf von TDL-Programmen befassen. Spätestens hier werden Sie dann aufgefordert Echtzeit-Kontrollanwendungen zu entwerfen.

Literatur

[1] Thomas A. Henzinger and Christoph M. Kirsch. The Embedded Machine: Predictable, Portable Real-Time code. Technical report, University of California at Berkeley, Berkely, USA, 2001.

[2] Christoph M. Kirsch. Principles of Real-Time Programming. *LNCS*, 2491, 2002.

[3] Stig Nielsson. Introduction to the legOS kernel, September 2000.

[4] Michael Haugaard Pedersen, Morten Klitgaard Christiansen, and Thomas Glaesner. Solving the Priority Inversion Problem in legOS. Technical report, University of Aalborg, May 2000.