Fachhochschule Würzburg-Schweinfurt

Wintersemester 2003/2004

Diplomfachprüfung im Fach

Prozessdatenverarbeitung II – Echtzeit-Programmierung

(Prof. Dr.-Ing. Ludwig Eckert)

Datum: 30. 01.2004

Dauer: 45 Minuten Erreichte Punktzahl: .....

Note: .....

Hilfsmittel: ohne schriftliche Unterlagen, Taschenrechner erlaubt

Vorname: ..... Name: .....

Matrikelnr: .....

Erstprüfer:.....

Zweitprüfer: .....

Wichtige Hinweise:

Bitte tragen Sie alle Antworten auf den folgenden Seiten direkt im Anschluss an die Aufgabentexte ein.

Ergebnisse auf Zusatzblättern werden nur in begründeten Ausnahmefällen gewertet.

Frage 1:

Nehmen Sie bitte eine Definition folgender Begriffe vor:

- Embedded System

- Echtzeit-Betriebssystem

Frage 2:

Geben Sie die Unterschiede zwischen einem harten und einem weichen „Real Time“-System an.

hartes „Real Time“-System	weiches „Real Time“-System

Frage 3:

Nennen Sie drei auf dem Markt verfügbare Echtzeit-Betriebssysteme.

Frage 4:

Der Einsatz von Echtzeit-Betriebssystemen in Embedded Anwendungen steigt derzeit rapide an. Geben Sie zwei mögliche Gründe für diese Entwicklung an.

Frage 5:

Der Einsatz von Echtzeit-Betriebssystemen in Embedded Anwendungen steigt derzeit rapide an. Geben Sie drei mögliche Gründe für diese Entwicklung an.

Frage 6:

Benennen Sie die Komponenten eines Minimal-Echtzeitbetriebssystems und erläutern Sie in knapper Form deren Funktion (Skizze).

Frage 7:

Zu welcher Gruppe von Systemprogrammen (Betriebssystem, Laufzeitsystem, Systemunterstützung) würden Sie die folgenden Programme/Programmteile rechnen?

a) ein Assembler

b) ein Programm, das die Hintergrund- bzw. die Arbeitsspeicherzuteilung vornimmt und automatisch Daten zwischen diesen Speichern austauscht

c) ein Programm, das den Namen einer Datei auf dem Hintergrundspeicher ändert

d) IP-Stack

e) Flash Disk-Emulation

f) Samba-Server

Aufgabe 8:

Gegeben ist das zeitliche Soll verhalten von 4 Rechenprozessen (RP1 - 4) dargestellt, wobei der RP4 die niedrigste und RP1 die höchste Priorität aufweist.

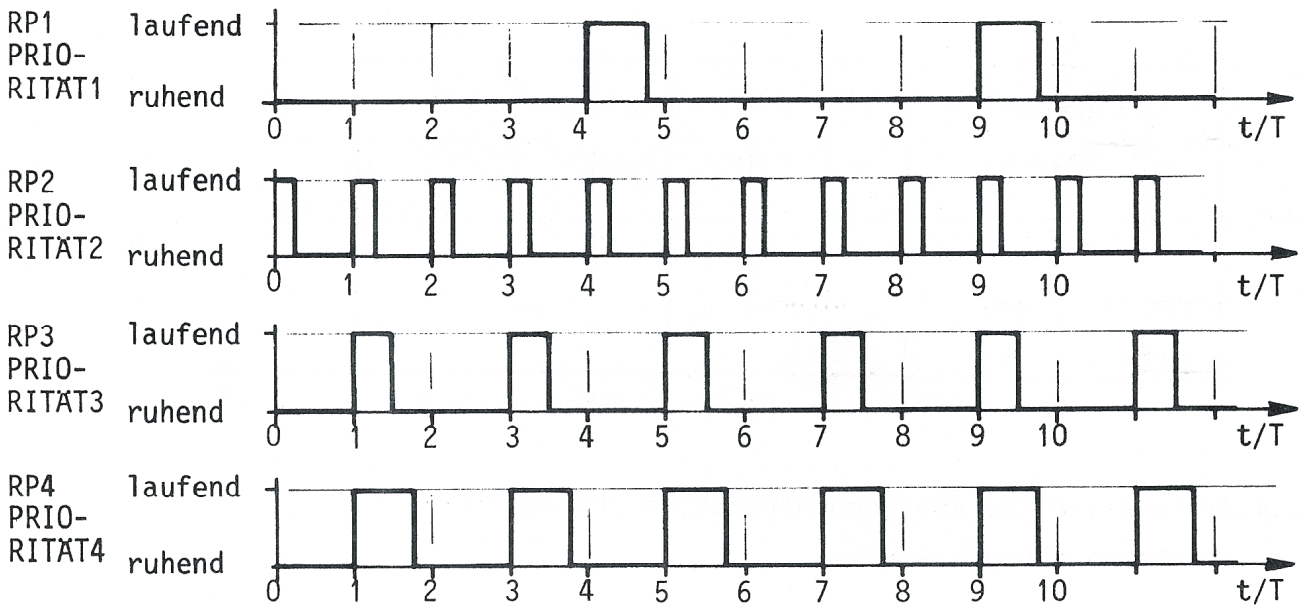

Bild: Zeitliches Sollverhalten der Rechenprozesse

a) Tragen Sie in das nachstehende Bild die tatsächliche Abarbeitung der Rechenprozesse ein (bis zum normierten Zeitpunkt t/T = 10).

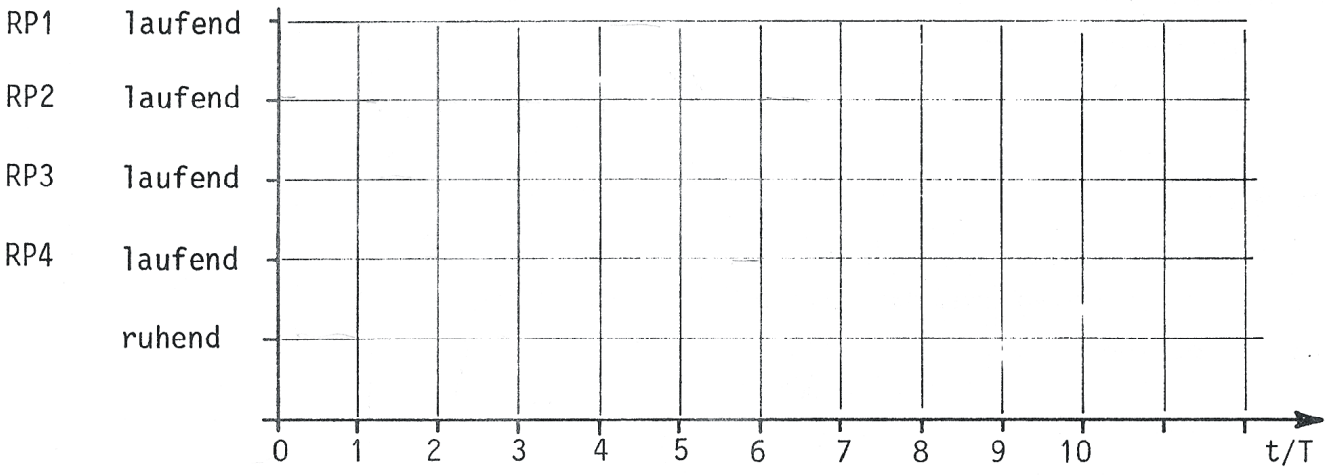

Bild: Zeitlicher Ablauf der Rechenprozesse

b) Wie beurteilen Sie die Ausführung der einzelnen Rechenprozesse RP1 bis RP4 unter dem Gesichtspunkt der Rechtzeitigkeit?

c) Tragen Sie den zeitlichen Verlauf der Taskzustände von Rechenprozess 4 in das nachstehende Bild ein.

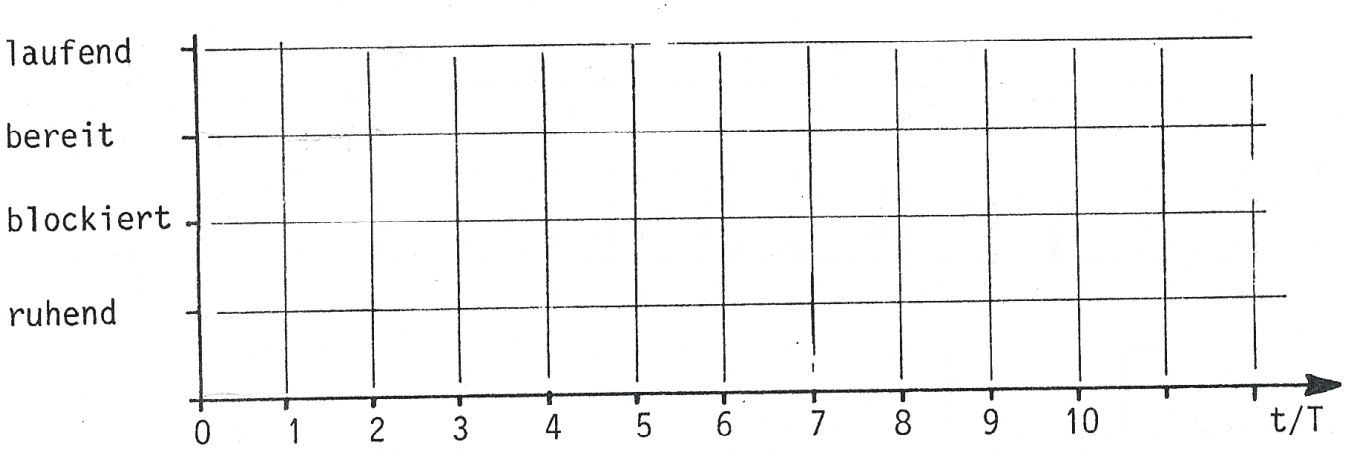

Bild: Taskzustände des Rechenprozesses RP4

Aufgabe 9:

In einem Echtzeit-Programm zur Automatisierung eines Produktionsprozesses werden 3 Teilprozesse durch jeweils eine Task überwacht. Die Tasks sollen im Folgenden mit ÜBERWACHENi bezeichnet werden, wobei i=1, 2, 3 sein soll.

Jede dieser Tasks liest spezifische Werte von dem jeweiligen Teilprozess ein. Anschließend werden diese Messwerte durch eine universelle Protokolliertask (die Protokolliertask trägt die Bezeichnung PROTOKOLL) in Form von Tabellen und Diagrammen auf einem Drucker ausgegeben.

Durch eine geeignete Synchronisierung soll folgender Ablauf der angeführten Tasks erzwungen werden:

- Nach jeder Überwachungstask soll einmal die Protokolliertask ablaufen.
- Die Reihenfolge der Überwachungstasks untereinander ist durch die Zahl in ihrem Namen festgelegt, d. h. zuerst ÜBERWACHEN1, ÜBERWACHEN2, ... Das Programm einer Task soll vollständig abgearbeitet sein, bevor das Programm einer anderen Task abgearbeitet wird.
- Der oben vorgegebene Ablauf soll zyklisch ablaufen können.

a) Schreiben Sie die Namen der Tasks in der oben geforderten Ablaufreihenfolge auf.

b) Führen Sie in jeder Task an der geeignenten Stelle die erforderlichen Synchronisierungsoperationen zur Sicherstellung der Ablaufreihenfolge ein. Verwenden Sie dabei ausschließlich binäre Semaphoren und die Befehle *SemGive(SemID)* und *SemTake(SemID)*, wobei für *SemID (=SemaphoreID)* beliebige Namen verwendet werden können, z. B. *SemID* = S1 für die Tasksynchronisation von Task ÜBERWACHEN1, *SemID* = S2 für die Tasksynchronisation von Task ÜBERWACHEN2 usw..

Tasks

ÜBERWACHEN1	ÜBERWACHEN2	ÜBERWACHEN3	PROTOKOLL
....	....	....	....
// C-Source // Überwachen1	// C-Source // Überwachen2	// C-Source // Überwachen3	// C-Source // Protokollieren
....	....	....	....
END.	END.	END.	END.

c) Die Semaphoren stellen unter VxWorks Kernelobjekte dar und müssen deshalb vor Ihrer Verwendung mit SemBCreate() einmalig erzeugt und mit einem Anfangswert (SEM_EMPTY = 0, SEM_FULL = 1) initialisiert werden. Wie müssen die von Ihnen verwendeten Semaphore-Variablen (SemID1, ..) initialisiert werden?