Ostfalia Hochschule für angewandte Wissenschaften Fakultät Fahrzeugtechnik Prof. Dr.-Ing. V. von Holt Institut für Fahrzeuginformatik und Fahrzeugelektronik	Modulprüfung Embedded Systems SS 2011 27.06.2011	Name:..... Vorname..... Matr.Nr.:..... Unterschrift.....

Zugelassene Hilfsmittel: **Keine**
Zeit: 60 Minuten

Punkte:

1	2	3	Summe

Prozente Klausur (50%)	Prozente Labor (50%)	Gesamtnote

Aufgabe 1 (12 Punkte) – Kurzfragen

a) (3 P) Erläutern Sie Bedeutung des Begriffs „Echtzeit“! Was versteht man unter einem Harten Echtzeitsystem“, was unter einem „Weichen Echtzeitsystem“? Nennen Sie je ein Beispiel!

b) (3 P) Was versteht man unter einem „Ereignisgesteuerten System“, was unter einem „Zeitgesteuerten System“?

c) (3 P) Geben Sie eine Definition für „Task“! Was versteht man insbesondere unter dem „Taskkontext“?

d) (3 P) Was unterscheidet Systeme mit Preemptiven bzw. Nicht-Preemptivem Multitasking voneinander?

Aufgabe 2 (18 Punkte) – Scheduling

Ein Regelungssystem verfügt zur Messwerterfassung über 2 Sensoren, die in unterschiedlichen Intervallen Messwerte zur Zustandserfassung des Systems liefern. Die Sensormesswerte sowie die Regelung sollen in jeweils eigenen Tasks ablaufen. Die folgende Tabelle enthält die Zykluszeiten sowie die Laufzeiten der einzelnen Tasks:

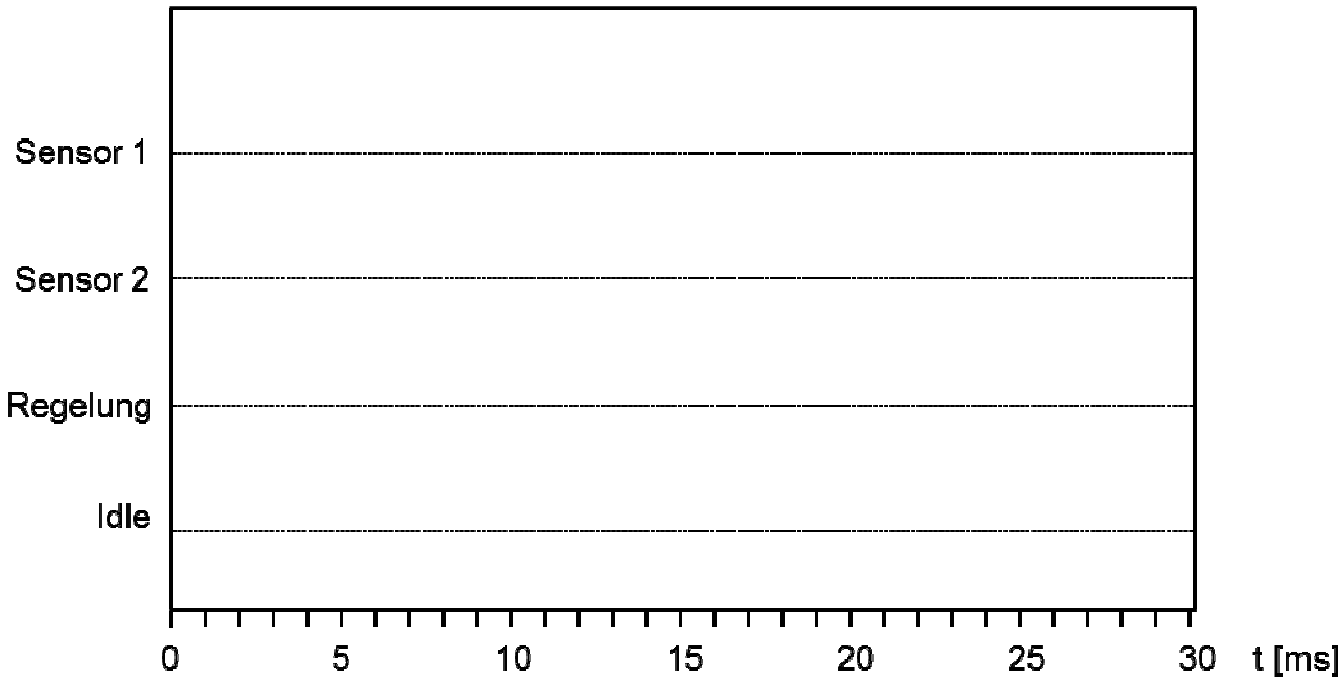
Tasks	Zykluszeit [ms]	Laufzeit[ms]
Sensor 1	5	2
Sensor 2	15	3…5
Regelung	10	2

(Die Deadline der Tasks entspricht der Periodendauer/Zykluszeit.)

a) (3 P) Berechnen Sie die Prozessorlast, die durch das Taskset verursacht wird! Ist das gegebene Taskset realisierbar?

b) (3 P) Welche Prioritäten müssen die Tasks erhalten, um nach dem Rate-Monotonic-Scheduling realisierbar zu sein? (Höchste Priorität : 0)

c) (9 P) Stellen Sie grafisch den Schedulingverlauf über eine Zeitdauer von 30ms im untenstehenden Diagramm dar! Gehen Sie dabei davon aus, dass zum Zeitpunkt t=0 für beide Sensoren Messwerte vorliegen und auch die Regelung ablaufbereit ist!



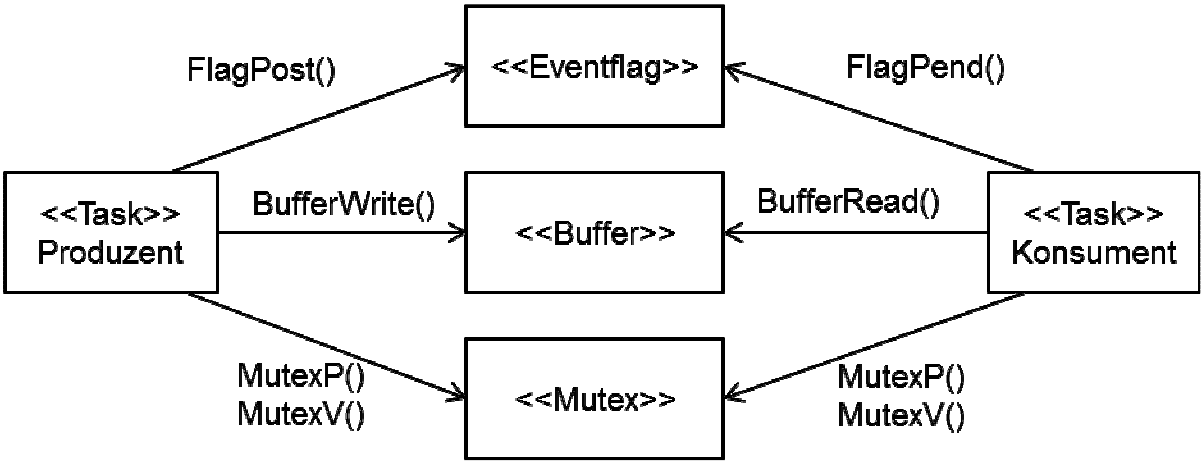
d) (3 P) Der Sensor 2 soll durch einen genaueren Sensor ersetzt werden. Dieser benötigt allerdings eine längere Auswertezeit. Wie lang darf die Auswertezeit höchstens werden, damit das System lauffähig bleibt?

Aufgabe 3 (20 Punkte) – Synchronisation/Kommunikation

a) (14 P) Zwischen einer „Produzenten“-Task und einer „Konsumenten“-Task sollen Daten über einen Puffer ausgetauscht werden. Der Zugriff auf den Puffer soll durch ein Mutex abgesichert sein. Um die Konsumenten-Task darüber zu informieren, dass Daten im Puffer vorhanden sind, steht ein Eventflag zur Verfügung, welches immer dann gesetzt sein soll, wenn Daten im Puffer vorhanden sind. Beim Zugriff auf den Puffer darf dieser sowohl vom Produzenten wie vom Konsumenten stets nur für **einen** Schreib- bzw. Lesevorgang belegt werden.

Folgende Funktionen stehen zur Verfügung:

```
void MutexP()       Passieren des Mutex. Task erhält Zugriff auf Puffer.
void MutexV()       Verlassen des Mutex. Task gibt Pufferzugriff frei.
void FlagPost()     Setzen des Eventflags.
void FlagPend()     Warten auf Eventflag. Im Anschluss ist das Eventflag
                    zurückgesetzt!
void BufferWrite()  Schreibt eine Nachricht in den Puffer.
int  BufferRead()   Liest eine Nachricht aus dem Puffer. Liefert die
                    Anzahl der noch im Puffer befindlichen Nachrichten
                    zurück.
```



Geben Sie den Ablauf des Zugriffs auf den gemeinsamen Puffer aus der Produzenten-Task und aus der Konsumenten-Task in Form eines Aktivitätsdiagramms oder in Form von Pseudocode an!

b) (6 P) In einem Echtzeitbetriebssystem können Nachrichten zwischen 2 Tasks über eine Messagequeue, die mehrere Nachrichten als „void-Pointer“ aufnehmen kann, ausgetauscht werden. Im folgenden Codebeispiel werden über Nachrichten Tastatureingaben zwischen 2 Tasks ausgetauscht. Leider enthält der Code einen folgenschweren Fehler, der einen fehlerfreien Ablauf behindert. Welcher Fehler ist dies?

```
// Task zum Einlesen von Tastatureingaben
void Look4InputTask(void)
{
  while (TRUE)
  {
    …
    if (…) // Taste gedrückt
      GetKey();
    …
  }
}

void GetKey(void)
{
  char c;

  c = … // Taste einlesen
  // Taste an HandleKeyCommandsTask senden
  SendMessage(KeyMailbox,&c);
}

// Task zum Verarbeiten von Tastatureingaben
void HandleKeyCommandsTask(void)
{
  char* pKeyMessage;
  char  KeyPressed;

  while (TRUE)
  {
    // Warten auf Nachricht von der Look4InputTask
    pKeyMessage = ReceiveMessage(KeyMailbox,WAIT_FOREVER);
    KeyPressed = *pKeyMessage;

    … // Taste verarbeiten
  }
}
```