


<p style="text-align: center;"><b>Ostfalia</b> Hochschule für angewandte Wissenschaften</p>  <p>Fakultät Fahrzeugtechnik Prof. Dr.-Ing. V. von Holt Institut für Fahrzeugsystem- und Servicetechnologien</p>	<p>Modulprüfung Embedded Systems BPO 2011</p> <p style="text-align: right;">WS 2014/15 07.01.2015</p>	<p>Name:.....</p> <p>Vorname.....</p> <p>Matr.Nr.:.....</p> <p>Unterschrift.....</p>
---	---	--

Zugelassene Hilfsmittel: **Einfacher Taschenrechner**  
Zeit: 60 Minuten

---

1 (12)	2 (22)	3 (26)	Summe (60)	Note

### Aufgabe 1 (12 Punkte) – Kurzfragen

- a) (4 P) Bei Echtzeitsystemen müssen Aktionen u.a. „**rechtzeitig**“ erfolgen. Was versteht man unter dem Begriff „**Rechtzeitigkeit**“ und welche **Varianten** der „**Rechtzeitigkeit**“ gibt es (Skizze)?
- b) (4 P) Was unterscheidet Systeme mit **Preemptiven** bzw. **Nicht-Preemptivem** Multitasking voneinander?
- c) (2 P) Wann bezeichnet man ein **Schedulingverfahren** als „**optimal**“?
- d) (2 P) Was versteht man unter einer „**Task**“ und was unter dem „**Taskkontext**“?

## Aufgabe 2 (22 Punkte) – Synchronisation/Kommunikation

Zwischen einer „Produzenten“-Task <<**Produzent**>> und einer „Konsumenten“-Task <<**Konsument**>> sollen Daten über einen Buffer in Form eines gemeinsamen Speichers ausgetauscht werden:

- Es gibt keine festgelegte Reihenfolge, in der die beiden Tasks auf den Speicher zugreifen.
- Die „Konsumenten“-Task soll informiert werden, wenn die „Produzenten“-Task etwas in den Speicher geschrieben hat.
- Die „Konsumenten“-Task soll eine Information über die Anzahl der hinterlegten Datensätze (= Anzahl der Schreibzugriffe der „Produzenten“-Task) erhalten.
- Beim Zugriff auf den Pufferspeicher darf dieser sowohl vom Produzenten wie vom Konsumenten stets nur für **einen** Schreib- bzw. Lesevorgang belegt werden.

Folgende Funktionen stehen zur Verfügung:

Kommunikationsmittel	Funktionen
Gemeinsamer Speicher (Buffer)	BufferWrite(), BufferRead()
Message Queue	MsgQPost(), MsgQPend()
Mutex	MuxPost(), MuxPend()
Semaphore	SemPost(), SemPend()
Event Flags	FlagPost(), FlagPend()

- a) (10 P) Entwerfen Sie eine Kommunikationsstruktur in UML-Notation, welche die o.g. Anforderungen umsetzt! Vermerken Sie an den Assoziationen der Tasks mit den Kommunikationsmitteln die jeweils benutzten Methoden (wie beim <<Buffer>>).



- b) (6 P) Stellen Sie den Ablauf der Speicherzugriffs von der <<Produzenten>>-Task auf den Buffer in einem **Aktivitätsdiagramm** oder in **Pseudocode** dar!
- c) (6 P) Stellen Sie den Ablauf der Speicherzugriffs von der <<Konsumenten>>-Task auf den Buffer in einem **Aktivitätsdiagramm** oder in **Pseudocode** dar!

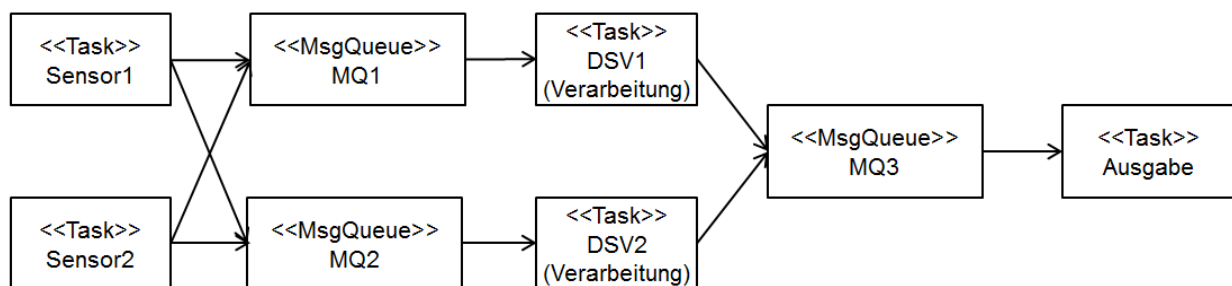
### Aufgabe 3 (26 Punkte) – Scheduling

Ein System zur Messwerterfassung von 2 Sensoren soll nach u.a. Skizze in Tasks strukturiert werden. Die Tasks kommunizieren über Message Queues miteinander.

Die **Sensor-Tasks** lesen die Messwerte gemäß der Zykluszeit laut u.a. Tabelle ein.

Die **Signalverarbeitungs-Tasks (DSV)** verarbeiten die in den Message Queues aufgelaufenen Messwerte und werden durch das Einfügen eines Werts in die Message Queue getriggert. Die in der Tabelle angegebene Laufzeit bezieht sich auf die Verarbeitung **eines** Messwerts. Liegen mehrere Messwerte vor, so skaliert die Laufzeit mit der Anzahl der Messwerte. Beachten Sie, dass die Sensor-Tasks die Messwerte sowohl an MQ1 wie MQ2 übergeben!

Die **Ausgabe-Task** übermittelt die verarbeiteten Messwerte zur Weiterverarbeitung an einen anderen Rechner.



Die folgende Tabelle enthält die Zykluszeiten sowie die Laufzeiten der einzelnen Tasks:

Tasks	Zykluszeit [ms]	Laufzeit[ms]
Sensor1	20	2
Sensor2	100	2
DSV1		2...4 / Messwert
DSV2		4...8 / Messwert
Ausgabe	100	10

- a) (3 P) Berechnen Sie die mittlere **Prozessorlast**, die durch das **Taskset** verursacht wird!

- b) (6 P) Berechnen Sie die **Prozessorlast pro Task**, die im **Worst-Case** auftreten kann sowie die daraus resultierende **Gesamtprozessorlast**!

Tasks	Prozessorlast [%]
Sensor1	
Sensor2	
DSV1	
DSV2	
Ausgabe	

- c) (2 P) Ist das gegebene Taskset grundsätzlich realisierbar?  
Ist es durch Rate-Monotonic-Scheduling realisierbar?

- d) (5 P) Das Taskset soll durch **Time-Slice-Scheduling** realisiert werden.  
Wählen Sie die Zeitscheiben je Task so, dass alle Zeitschranken eingehalten werden (Sensor-Tasks und Ausgabe-Tasks) und nach 100 ms auch alle Sensorwerte verarbeitet sind.  
Gehen Sie davon aus, dass die Task-Reihenfolge beim Time-Slicing wie folgt ist:  
**Sensor1 – Sensor2 – DSV1 – DSV2 – Ausgabe.**  
Tragen Sie die gewählten Zeitscheiben in die Tabelle ein!

Tasks	Zeitscheibe [ms]
Sensor1	
Sensor2	
DSV1	
DSV2	
Ausgabe	

- e) (10 P) **Alternativ soll das Taskset durch ein Earliest-Deadline-First-Scheduling** realisiert werden. Weisen Sie den Schedule für den **Worst-Case** anhand des untenstehenden **Schedulediagramms** nach! Nehmen Sie für die beiden DSV-Tasks vereinfachend als Deadline die Zykluszeit von Sensor1 an.

