Ostfalia Hochschule für angewandte Wissenschaften Fakultät Fahrzeugtechnik Prof. Dr.-Ing. V. von Holt Institut für Fahrzeugsystem- und Servicetechnologien	Modulprüfung Embedded Systems BPO 2008 WS 2013/14 15.01.2014	Name:..... Vorname..... Matr.Nr.:..... Unterschrift.....

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

Echtzeitbetriebssysteme:

1 (8)	2 (34)	3 (18)	Summe (60)	Note

Embedded Systems:

Ausarbeitung (50%)	Labor (50%)	Summe	Note

Aufgabe 1 (8 Punkte) – Kurzfragen

a) (4 P) Was versteht man unter einem „**Ereignisgesteuerten System**“, was unter einem „**Zeitgesteuerten System**“?

b) (4 P) Was unterscheidet Systeme mit **Preemptiven** bzw. **Nicht-Preemptivem** Multitasking voneinander?

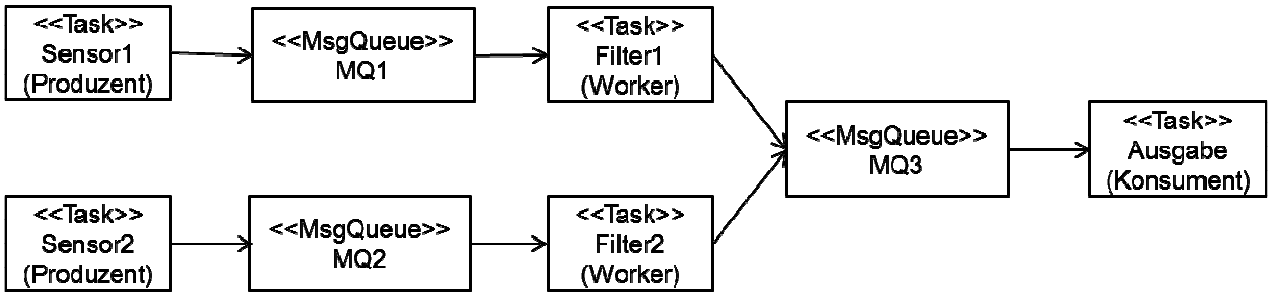
Aufgabe 2 (34 Punkte) – Scheduling

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 **interruptgesteuert** die Messwerte ein, deren Eintreffen innerhalb der angegebenen Zeiten (s. Tabelle) **schwankt**. Im Mittel über längere Zeiträume von einigen Sekunden gleichen sich die Schwankungen aus.

Die **Filter-Tasks** werden **timergesteuert** periodisch aktiviert und verarbeiten in einem Durchlauf die in den Message Queues aufgelaufenen Messwerte. 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. Die Filter-Tasks verarbeiten in einem Durchlauf nur die bei ihrer **Aktivierung** in der **Message-Queue befindlichen Messwerte**!

Die **Ausgabe-Task** übermittelt die gefilterten 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	10..30	1
Sensor2	20..30	2
Filter1	20	2…4 / Messwert
Filter2	25	3…5 / Messwert
Ausgabe	100	8

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

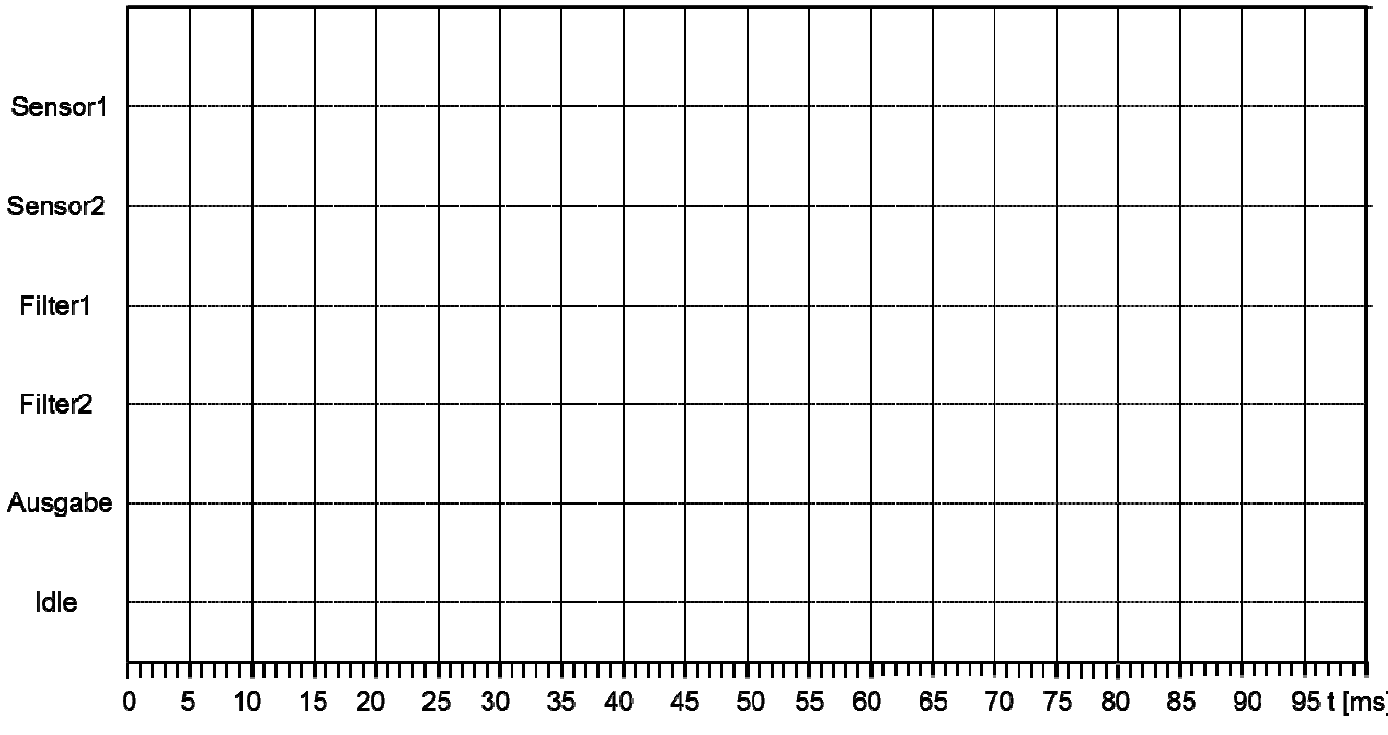
a) (2 P) Berechnen Sie die mittlere **Prozessorlast**, die durch das **Taskset** verursacht wird!

b) (5 P) Berechnen Sie die **Prozessorlast**, die im **Worst-Case** durch das **Taskset** verursacht wird! Tragen Sie dazu in der u.a. Tabelle, die von Ihnen getroffenen Annahmen für den Worst-Case ein! Ist das gegebene Taskset **realisierbar**?

Tasks	Zykluszeit [ms]	Laufzeit[ms]
Sensor1		
Sensor2		
Filter1		
Filter2		
Ausgabe		

c) (5 P) Das Taskset soll durch ein **Rate-Monotonic-Scheduling** realisiert werden soll.
Welche **Prioritäten** müssen den **Tasks** jeweils zugewiesen werden? (**Höchste Priorität : 0**)
Nach welcher **Regel** werden die **Prioritäten vergeben**?
Ist das **Taskset** in jedem Fall mit RMS-Scheduling **umsetzbar** (Begründung)?

d) (12 P) Weisen Sie den Schedule für den **Worst-Case** anhand des untenstehenden **Scheduldiagramms** nach! Gehen Sie dabei davon aus, dass zum **Zeitpunkt t=0 alle Tasks lauffähig** sind bzw. alle **Ereignisse eintreffen** und die **Message Queues leer** sind!



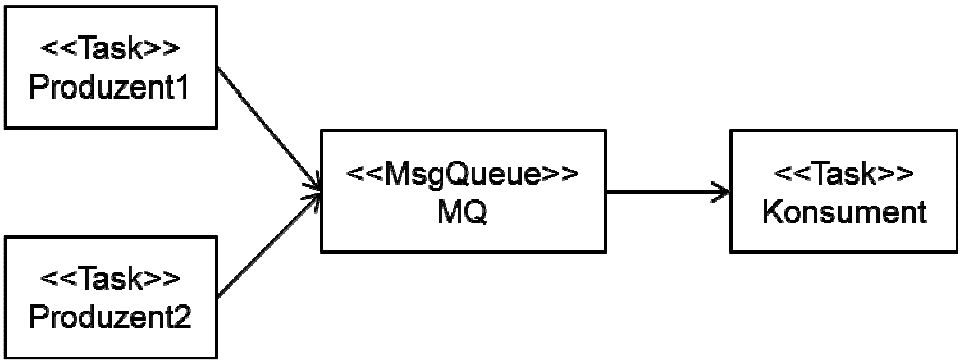
e) (3 P) Wie groß müssen die 3 Message Queues sein, d.h. über wie viele Speicherplätze müssen diese verfügen?

f) (3 P) Wie groß müssten die 3 Message Queues sein, wenn die beiden Filter-Tasks nur mit einer Periode von 50ms laufen?

g) (4 P) Alternativ wird darüber nachgedacht, die beiden Filter-Tasks **nicht periodisch** zu starten, sondern **ereignisgetriggert** immer dann, wenn mindestens 2 Botschaften in den Message Queues liegen. Um welches Element müsste das Strukturbild ergänzt werden und wie wäre damit das Zusammenspiel der Sensor- und der Filter-Tasks?

Aufgabe 3 (18 Punkte) – Synchronisation / Kommunikation

Ähnlich der Aufgabe 2 wird die Kommunikation zwischen 2 Produzenten-Tasks und 1 Konsumenten-Task durch eine Message Queue realisiert. Da die Verarbeitungsdauer des Konsumenten schwankt, schwankt ebenso die Anzahl der Nachrichten in der Message Queue. Es soll in jedem Fall verhindert werden, dass Nachrichten der Konsumenten verlorengehen durch eine belegte Message Queue. Hierzu ist eine sogenannte Flusskontrolle vorzusehen, die dafür sorgt, dass die Produzenten nur dann eine Nachricht an den Konsumenten schicken, wenn die Message Queue nicht voll belegt ist.



a) (6 P) Erweitern Sie das o.a. Strukturbild um Elemente, die eine solche Flusskontrolle ermöglichen! Das Betriebssystem stellt Ihnen folgende Elemente zur Verfügung:

- Mutex (MuxPost(), MuxPend())
- Semaphore (SemPost(), SemPend())
- MsgQueue (MsgQPost(), MsgQPend())

Beachten Sie, dass Message Queue über keine Funktionen verfügt, die eine Abfrage des aktuellen Füllstands zulassen!

b) (12 P) Wie sieht der Zugriff und das Management der Message Queue auf Seiten der Produzenten- und der Konsumenten-Tasks aus?
Stellen Sie beides als Pseudo-Code dar!