


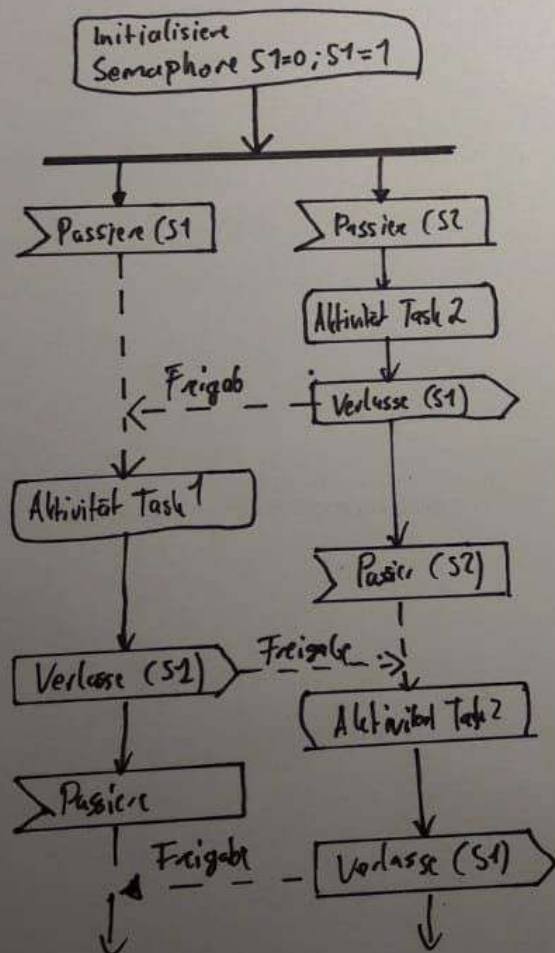
<b>Ostfalia</b> Hochschule für angewandte Wissenschaften  Fakultät Fahrzeugtechnik Prof. Dr.-Ing. V. von Holt Institut für Fahrzeugsystem- und Servicetechnologien	 Modulprüfung Embedded Systems BPO 2011/BPO 2008  WS 2015/2016 06.01.2016	Name:..... Vorname:..... Matr.Nr.:..... Unterschrift:.....
---	---	---

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

1 (16)	2 (32)	3 (12)	Summe (60)	Note

- e) (8 P) Erläutern Sie anhand eines Aktivitätsdiagramms das Prinzip der Reihenfolgesynchronisation mit Semaphoren beim Zugriff auf geteilte Ressourcen!

VL 6 S. 46



- a) (3 P) Welche Annahmen nach o.a. Tabelle charakterisieren den **Worst-Case** bzw. den **Best-Case**?

Worst Case: kürzeste Zykluszeit; längste Laufzeit

Best Case: längste Zykluszeit; kürzeste Laufzeit

- b) (4 P) Bestimmen Sie die **mittlere Zykluszeit** der SV-Task sowie der SO-Task für den Worst-Case und den Best-Case!

~~$t_{\text{em}} = 20 + 30 + 50 = 100 \Rightarrow 33,33 \text{ ms}$~~  SV und SO gleiche  $t_{\text{Zyklus}}$   
 Best ~~best~~: kleinste gemeinsame Vielfache:  $S_1, S_2, S_3 = 120 \text{ ms} = 4 \cdot S_1, 3 \cdot S_2, 2 \cdot S_3 \Rightarrow \frac{120 \text{ ms}}{4+3+2} = 17,3 \text{ ms}$   
 Worst ~~best~~: 11  $\Rightarrow$  KGV  $= 40 \text{ ms} \Rightarrow 4 \cdot S_1; 2 \cdot S_2; 7 \cdot S_3 \Rightarrow \frac{40 \text{ ms}}{4+2+7} = 5,71 \text{ ms}$

- c) (6 P) Berechnen Sie die minimale, maximale sowie die mittlere **Prozessorlast**, die durch die einzelnen Tasks sowie durch das gesamte **Taskset** verursacht wird!

Tasks	Minimal [%]	Maximal [%]	Gemittelt [%]
SI1	3,3	10	5
SI2	2,5	5	3,3
SI3	3,3	5	4
SV	$\frac{1}{3} \cdot x_1$	$\frac{1}{2} \cdot x_2$	$x_3$
SO	$\frac{1}{4} \cdot x_3 = 2,5$	$\frac{1}{2} \cdot x_2 = 12,5$	$x_4$
Gesamt	42,4	90	54,6

Laufzeit  
 Zykluszeit  
 $x_1 = \frac{4 \cdot 2 + 3 \cdot 3 + 2 \cdot 7}{120 \text{ ms}} = 25,83\%$   
 $x_2 = \frac{4 \cdot 2 + 3 \cdot 2 + 1 \cdot 7}{40 \text{ ms}} = 52,5\%$   
 $x_3 = \frac{15 \cdot 2 + 10 \cdot 3 + 6 \cdot 2}{300 \text{ ms}} = 34\%$

- d) (1 P) Ist das gegebene Taskset nach den Berechnungen unter c) prinzipiell realisierbar?

Ja da nicht über 100% wäre möglich

- e) (6 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)?

Regel:  $\text{Priorität}(\text{Task}_i) \sim \frac{1}{\text{Periodendauer}}$

theoretische Obergrenze der Auslastung:  $H = n(2^{\frac{1}{n}} - 1) \xrightarrow{n=\text{Anzahl Tasks}} H = 5 \cdot (2^{\frac{1}{5}} - 1) = 4,36$

Es ist nicht in jedem Fall umsetzbar da die Max. Auslastung bei 90% liegt

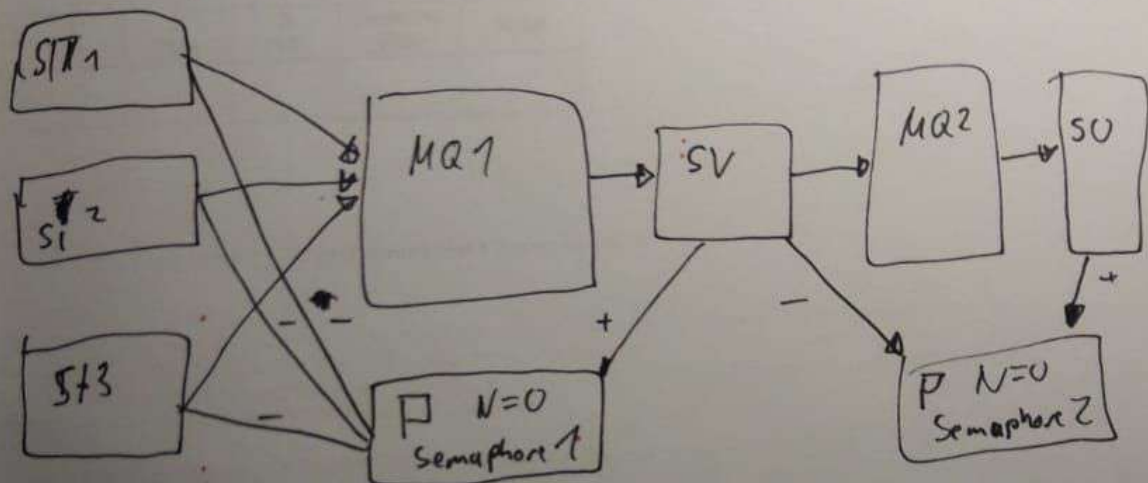
- f) (12 P) Weisen Sie anhand des u.a. Schedulingdiagramms für den **Worst-Case** nach, ob das Taskset mittels **RMS** realisierbar ist!  
 (Die Message Queues sind zum Startzeitpunkt als leer anzunehmen!)

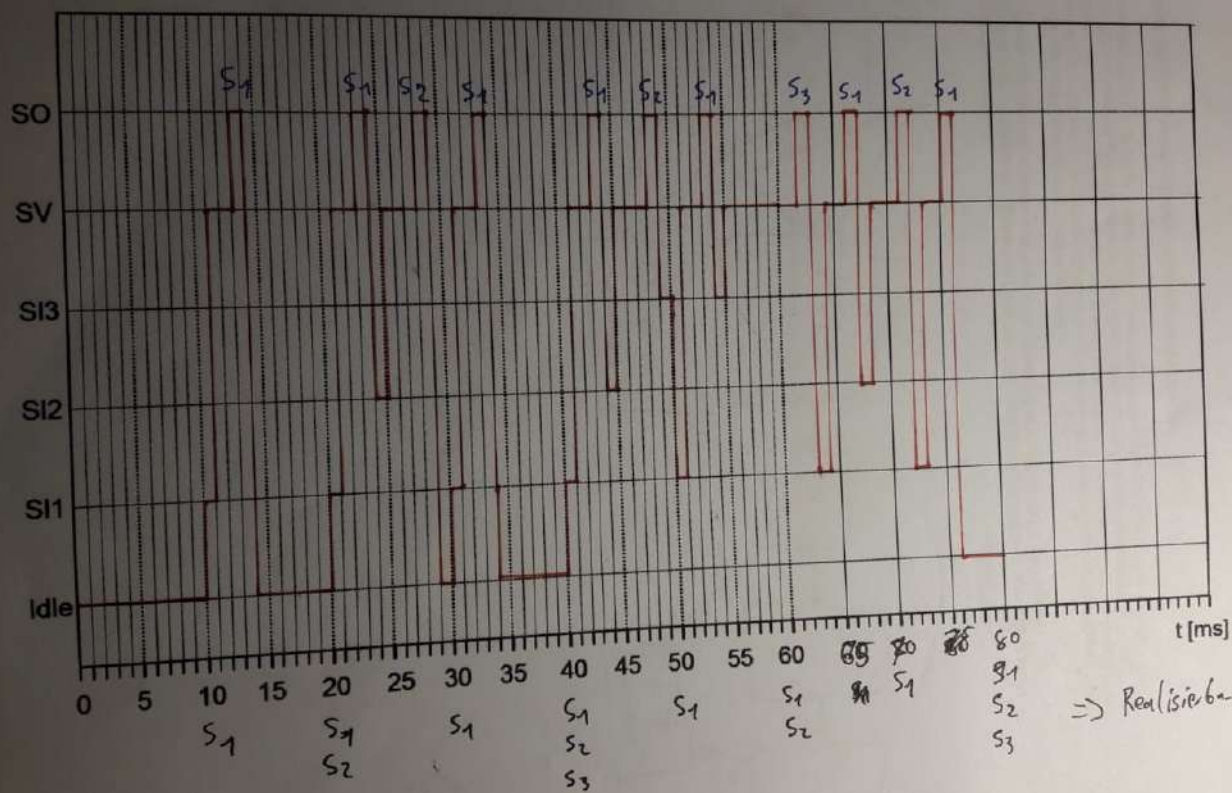


### Aufgabe 3 (12 Punkte) – Synchronisation/Kommunikation

Ein Nachteil der in Aufgabe 2 verwendeten Taskanordnung besteht darin, dass unter ungünstigen Umständen eine der Message Queues überlaufen könnte, d.h. es könnten Informationen verloren gehen. Um dem vorzubeugen soll die Anordnung um eine sogenannte Flusskontrolle erweitert werden, bei der die in die Message Queues schreibenden Tasks nur dann eine Information weitergeben, wenn in den Message Queues noch Platz vorhanden ist.

Ergänzen Sie den u.a. Ausschnitt aus der Taskanordnung von Aufgabe 2 um eine solche Flusskontrolle und erläutern Sie deren Funktionsweise anhand von Pseudocode oder mittels eines Aktivitätsdiagramms!





=> Realisierbar

### Aufgabe 1 (16 Punkte) – Kurzfragen

a) (2 P) Welche 2 wesentlichen Aufgaben erfüllt ein (allgemeines) „Betriebssystem“?

- Verbindung von Hardware und Anwendungsprogrammen

- „Scheduling“ / Verwaltung der Programmabläufe  
(Zuweisung von Rechenzeit)

b) (2 P) Was unterscheidet Systeme mit preemptiven bzw. nicht-preemptivem Multitasking voneinander?

? Vorlesung

preemptiv: Das Betriebssystem entscheidet welche Task wann und wie lange läuft

nicht preemptiv: Die laufende Task entscheidet wann der Proz. ~~weiter~~ freigegeben wird.

c) (2 P) Wann bezeichnet man ein Schedulingverfahren als „optimal“?


Optimal wenn alle möglichen Schedules als diese erkannt werden.

d) (2 P) Was versteht man unter einer „Task“ und was unter dem „Taskkontext“?

Task: Ausführbares Programm  
Aktionsfunk. + Zustandsvariablen

Taskkontext: Zustandsvariablen

{	Programm Zähler
	Register
	Stack
	Variablen

<b>Ostfalia</b> Hochschule für angewandte Wissenschaften   Fakultät Fahrzeugtechnik Prof. Dr.-Ing. V. von Holt Institut für Fahrzeugsystem- und Servicetechnologien	Modulprüfung Embedded Systems BPO 2011  WS 2016/17 06.01.2017	Name:..... Vorname:..... Matr.Nr.:..... Unterschrift:.....
---	--	---

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

1 (8)	2 (38)	3 (14)	Summe (60)	Note



### Aufgabe 1 (8 Punkte) – Kurzfragen

- a) (2 P) Was unterscheidet das Versetzen einer Task in den **Wait-Zustand** von dem Aufruf einer „klassischen“ Delay-/Wait-Funktion?

Im Wait Zustand einer Task kann der Prozessor weiterhin andere Tasks abarbeiten.  
Beim „klassischen“ Wait passiert dies nicht.

- b) (2 P) Warum darf in Multitaskingumgebungen eine Task **nicht ständig aktiv** sein?

Weil andere Task nicht aktiv sein können.

- c) (2 P) Wann bezeichnet man ein **Schedulingverfahren** als „optimal“?

Wenn alle existierenden Schedules gefunden wurden.

- d) (2 P) Was versteht man unter einer „Task“ und was unter dem „Taskkontext“?

Task = Programmablauf für verschiedene Aufgaben  
Aktionsfunktionen + Zustandsvariablen

Taskkontext =  $\left\{ \begin{array}{l} \text{Programnzähler} \\ \text{Register} \\ \text{Stack} \\ \text{Variablen} \end{array} \right.$  = Zustandsvariablen



Das Taskset soll im Folgenden durch ein Least-Laxity-First-Scheduling realisiert werden. Hierbei sollen  
 – abweichend zu oben – die folgenden Annahmen über die Zyklus- und Laufzeiten der Tasks gelten:

Tasks	Zykluszeit [ms]	Laufzeit[ms]	l
T1	5	1	4
T2	10	2	8
T3	10	3	7
T4	15	4	11

- d) (16 P) Geben Sie in **Tabelle 1** für den Zeitraum 0...30ms für alle Tasks die jeweilige **Restlaufzeit** sowie den **Spielraum** (laxity) an! Hierbei soll vereinfachend angenommen werden, dass das Betriebssystem **nur alle Millisekunde einen Taskwechsel** durchführt. Markieren Sie in jedem Zeitschritt durch **Einkreisen der Zeiten**, welche **Tasks** im jeweiligen Zeitschritt **abläuft**!

$$l_i = d_i - (t + e_{r_i})$$

- e) (8 P) Stellen Sie den **Verlauf des Spielraums** (laxity) der Tasks in **Diagramm 1** für den Zeitraum 0...15ms grafisch dar! Geben Sie **farblich** oder durch **Annotationen** an, welcher **Tasks** die jeweiligen **Kurven entsprechen**!

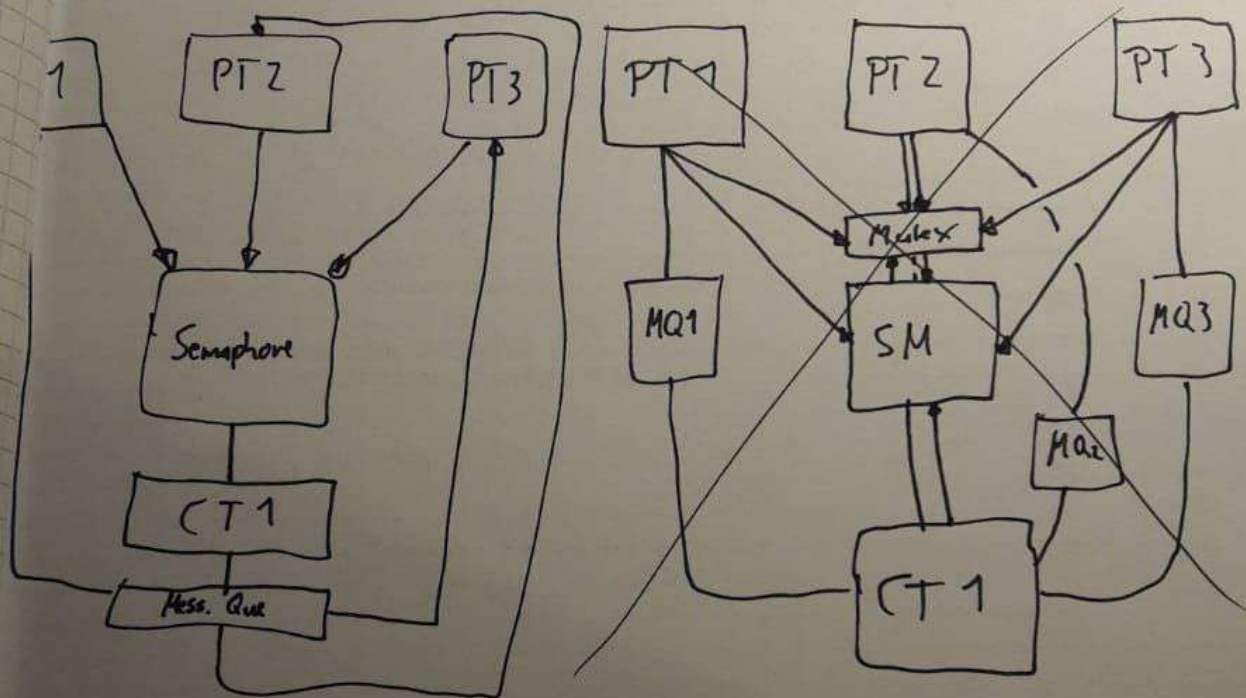
### Aufgabe 3 (14 Punkte) – Flusskontrolle

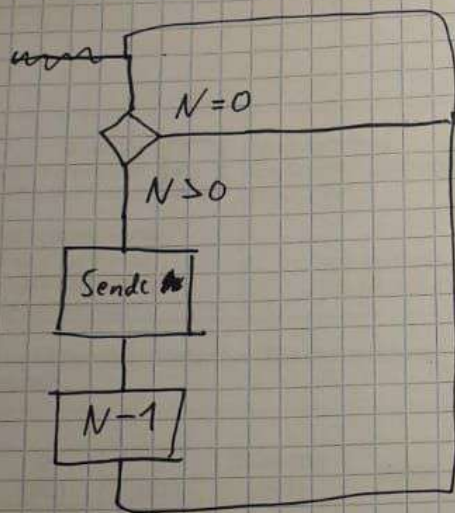
3 **Producer-Tasks** sollen 1 **Consumer-Tasks** Daten zur Weiterverarbeitung übermitteln. Damit die Consumer-Tasks nicht mit Daten überflutet wird bzw. keine Daten durch Pufferüberläufe verlorengehen, soll zwischen den Producer-Tasks und der Consumer-Tasks eine **Flusskontrolle** implementiert werden. Die maximale Anzahl der zu puffernden Datensätze soll **N** betragen.

Anmerkung: Das Betriebssystem stellt Ihnen folgende Kommunikations-/Synchronisationskonstrukte zur Verfügung: **Message Queue, Mailbox, Semaphore, Mutex, Shared Memory**

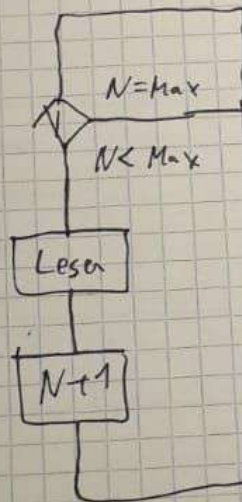
a) (6 P) Stellen Sie grafisch eine mögliche Lösung für eine Flusskontrolle dar.

b) (8 P) Erläutern Sie die Funktionsweise Ihrer gewählten Anordnung anhand von Pseudocode oder eines Aktivitätsdiagramms!





(Producer)



(Consumer)



	$T_{L1}$	$T_{R1}$	$T_{L2}$	$T_{R2}$	$T_{L3}$	$T_{R3}$	$T_{L4}$	$T_{R4}$
0	4	1	8	2	7	3	11	4
1	4	0	2	2	6	3	10	4
2			6	2	6	2	9	4
3			5	2	6	1	8	4
4			5	1	5	1	7	4
5	4	1	5	0	4	1	6	4
6	4	0			3	1	5	4
7					3	0	4	4
8							4	3
9							4	2
10	4	1	8	2	7	3	4	1
11	4	0	7	2	6	3		
12			6	2	5	3	4	0
13			5	2	5	2		
14				2	5	1		
15	4	1	4	1	4	1	11	4
16	4	0	3	1	3	1	10	4
17			3	0	2	1	9	4
18					2	0	8	4
19							8	3
20			8	2	7	3	8	2
21			2	2	7	2	7	2
22								
23								
24								
25								
26								
27								

Nochmal Neu



## Aufgabe 2 (38 Punkte) – Scheduling

Zur Steuerung eines Echtzeitsystems ist ein Taskset bestehend aus 4 Tasks (T1, T2, T3, T4) entworfen worden.

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

Tasks	Zykluszeit [ms]	Laufzeit[ms]	Prio
T1	4...6	1	0
T2	8...10	1...2	1
T3	10...20	2	3
T4	15	3	2

$$L = \frac{t_c}{t_z}$$

- a) (8 P) Welche Annahmen nach o.a. Tabelle charakterisieren den **Worst-Case**, den **Best-Case** bzw. den **gemittelten** Wert der **Prozessorlast** durch die **einzelnen Tasks** sowie durch das **gesamte Taskset**?

Tasks	Minimal [%]	Maximal [%]	Gemittelt [%]
T1	16,6	25	20
T2	10	25	17,5
T3	10	20	15
T4	20	20	20
Gesamt	56,6	30	72,5

- b) (1 P) Ist das gegebene Taskset nach den Berechnungen unter a) prinzipiell realisierbar?


Ja

nicht über 100%

- 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** mit **RMS-Scheduling** **umsetzbar** (Begründung)?

$$\text{Priorität (Task } i) \sim \frac{1}{\text{Periodendauer (Task } i)}}$$

> Nicht umsetzbar da die Zykluszeit nicht bei allen Tasks konstant ist.

<b>Ostfalia</b> Hochschule für angewandte Wissenschaften  Fakultät Fahrzeugtechnik Prof. Dr.-Ing. V. von Holt Institut für Fahrzeugsystem- und Servicetechnologien	 Modulprüfung Embedded Systems BPO 2011  WS 2017/18 05.01.2018	Name:..... Vorname..... Matr.Nr.:..... Unterschrift.....
---	---	---

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

1 (8)	2 (30)	3 (22)	Summe (60)	Note

Aufgabe 1 (8 Punkte) – Kurzfragen

a) (2 P) Wann bezeichnet man ein Schedulingverfahren als „optimal“?

⇒ wenn alle existierenden Schedules gefunden werden.

⇒ <sup>Vorl.</sup> Folie 7 Seite 33

b) (2 P) Welche 2 wesentlichen Aufgaben erfüllt ein (allgemeines) „Betriebssystem“?

- Abstraktion von Hardware
- Multitasking
- Taskverwaltung
- Betriebsmittelverwaltung
- Synchronisation

c) (2 P) Was versteht man unter einer „Task“ und was unter dem „Taskkontext“?

Task = Ausgeführte Instanz eines Programms  
Aktionsfunktion + Zustandsvariablen

Taskkontext {   
    Programmzähler  
    Register  
    Stack  
    Variablen  
    } = Zustandsvariablen

d) (2 P) Was unterscheidet Systeme mit preemptiven bzw. nicht-preemptivem Multitasking voneinander? 2 504

Preemptiv : Task mit höherer Priorität verdrängen  
Tasks mit niedriger Priorität

Preemptiv : Task mit höherer Priorität starten  
erst sobald die vorherige Task abgelaufen

d) (2 P) Mit welcher Maßnahme kann eine Prioritäteninversion vermieden werden? (Erläuterung)

Durch Prioritätenvererbung

e) (10 P) Stellen Sie den Schedulingverlauf für das unter Teilaufgabe c) gegebene Taskset erneut in einem Zeitdiagramm dar, wobei Sie dieses Mal die unter Teilaufgabe d) genannte Maßnahme unterstellen!



Task 1

Task 2

Task 3

Task 4

Idle

zu Aufgabe 2 e)

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 [ms]

- $\triangle$  Mutex lock
- $\nabla$  Mutex unlock
- $\blacktriangle$  Mutex anfordern

$t_{T4}$

$t_{T3}$

$t_{T1}$

$t_{T2}$

$t_{T1}$

$t_{T1}$

b) (12 P) Erläutern Sie die Funktionsweise Ihrer gewählten Anordnung anhand von Pseudocode oder eines Aktivitätsdiagramms für jede der 3 Tasks!

Task 1:

While (1)

Warte auf Sensormessung()  
Mutex 12. lock()  
Schreibe Daten in SM12()  
Mutex 12. unlock()  
Sende Event 12()

Task 2:

While (1)

Warte auf Event 12()  
Mutex 12. lock()  
Lese Daten aus SM12()  
Mutex 12. unlock()  
Verarbeite Daten()  
Mutex 23. lock()  
Schreibe Daten in SM23()  
Mutex 23. unlock()  
Sende Event 23()

Task 3:

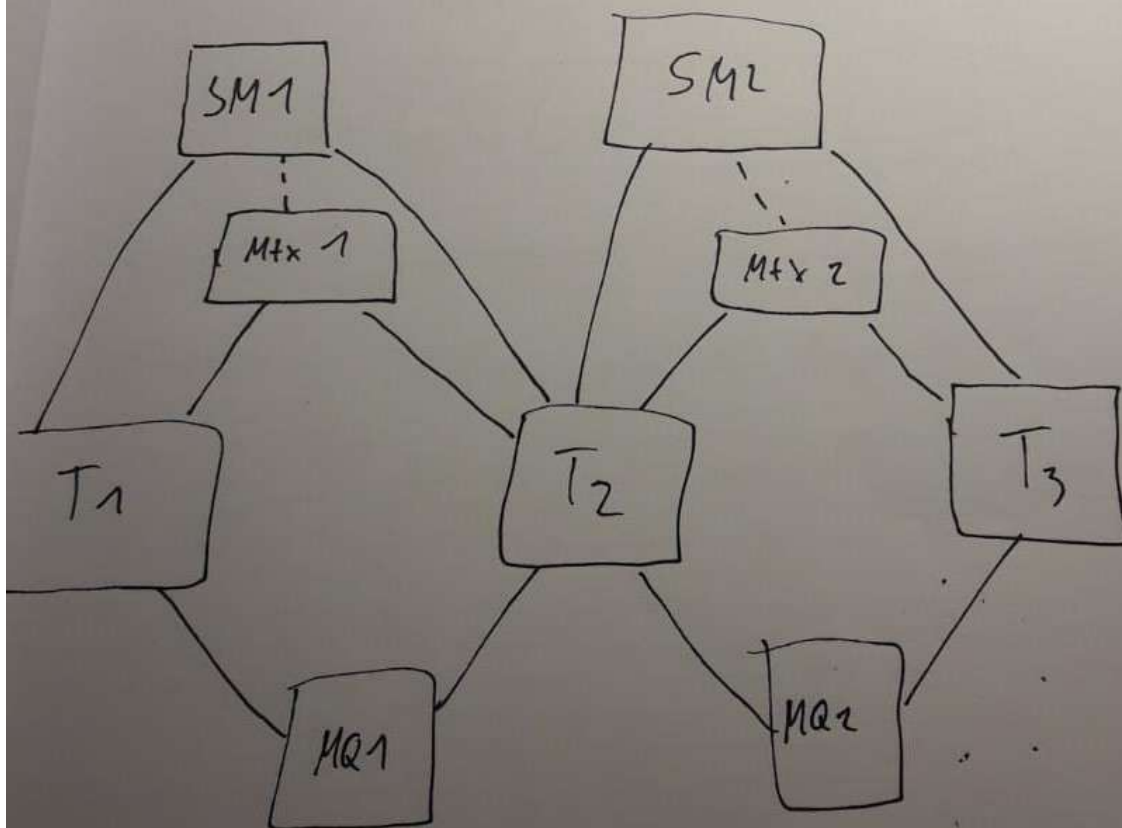
While (1)

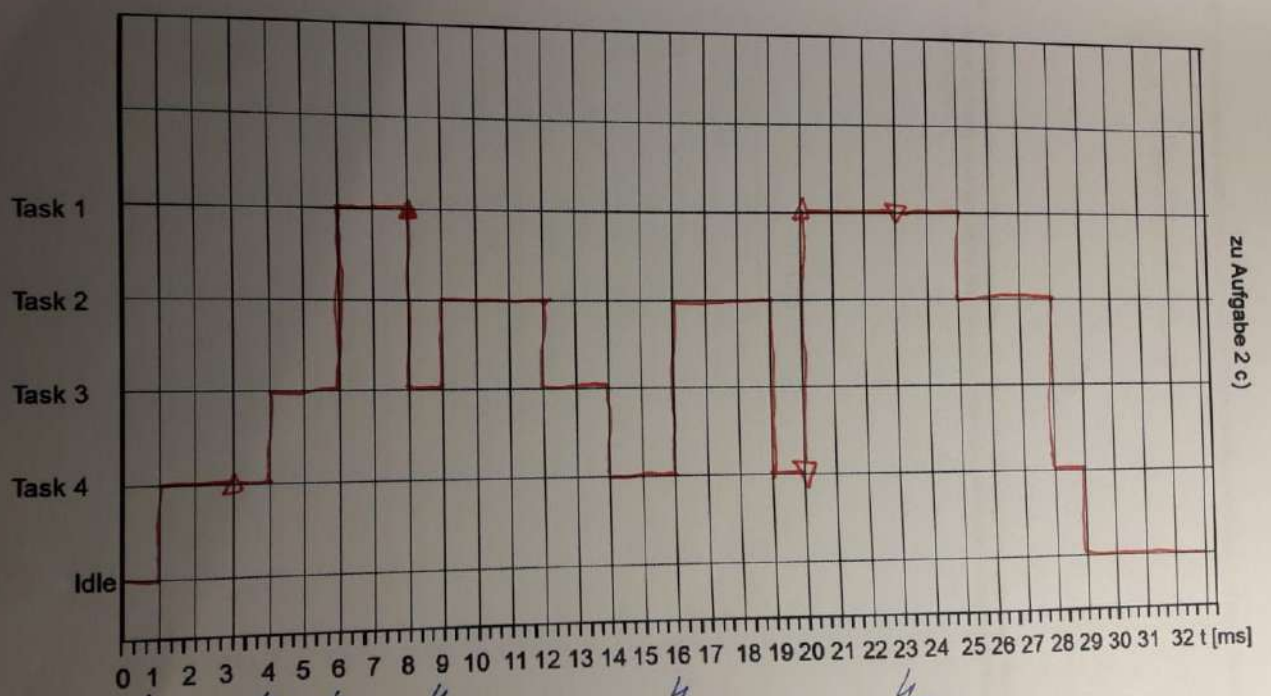
Warte auf Event 23()  
Mutex 23 lock()  
Lese Daten aus SM23()  
Mutex 23. unlock()  
Verarbeite Daten()

**Aufgabe 3 (22 Punkte) – Task-Kommunikation/Synchronisation**

3 Tasks  $T_1$ ,  $T_2$ ,  $T_3$  sollen Daten in Form einer Pipeline  $T_1 \Rightarrow T_2 \Rightarrow T_3$  verarbeiten. Der Datenaustausch zwischen den Tasks soll jeweils über ein **Shared Memory** erfolgen. Die Tasks sollen nach Bearbeitung und Weitergabe der Daten jeweils **passiv** auf neue Daten der Vorgängertask warten. (Bei Task 1 können Sie hierfür allgemein „Warte auf Sensordaten()“ annehmen.)

- a) (10 P) Stellen Sie grafisch eine mögliche Lösung für die o.a. Kommunikation der 3 Tasks dar. Nutzen Sie dazu nur die Ihnen bekannten Kommunikations-/Synchronisations-Konstrukte wie Message Queue, Mailbox, Semaphore, Mutex und Shared Memory.





zu Aufgabe 2 c)

Δ Mutex Lock  
 ▽ Mutex unlock



## Aufgabe 2 (30 Punkte) – Scheduling / Prioritäteninversion

Beim prioritätsbasierten Multitasking kann durch den Zugriff auf gemeinsame Ressourcen das Problem einer Prioritätsinversion auftreten.

a) (2 P) Was versteht man unter dem o.a. Begriff der Prioritätsinversion? (Semaphoren)

ohne Priorität: können Nachrichten von Tasks nied. Prio  
Nachrichten von Tasks höher Prio blockieren

mit Priorität: Nachrichten höher Priorität Tasks haben Vorrang

b) (4 P) Was versteht man unter einem Deadlock und einem Livelock?

Deadlock: Mehrere Tasks warten auf Freigabe von Betriebsmitteln,  
diese blockieren sich gegenseitig

Livelock: Task können werden durch Konspiration anderer Tasks  
ständig an Ausführung gehindert.

c) (12 P) Gegeben sei ein Taskset mit 4 Task mit zugeordneten Prioritäten und Ausführungszeiten entsprechend nachfolgender Tabelle:

Task	Priorität (1 = höchste Priorität)	Ausführungszeit [ms]
Task 1	1	2 - Mutex.Lock - 3 - Mutex.Unlock - 2
Task 2	2	3
Task 3	3	5
Task 4	4	2 - Mutex.Lock - 4 - Mutex.Unlock - 1

Gegeben sei ferner die folgende Ereignisfolge:

Ankunftszeit t [ms]	Task
1	Task 4
4	Task 3
6	Task 1
9	Task 2
16	Task 2
23	Task 2

Tragen Sie im nachstehenden Zeitdiagramm zunächst die Ereignisfolge an der Zeitachse auf! Stellen Sie im Anschluss den Schedulingverlauf dar, wie er sich bei Vorliegen einer Prioritäteninversion, d.h. ohne Gegenmaßnahmen, ergibt! Kennzeichnen Sie dabei die Zeitpunkte zu denen Mutex.Lock bzw. Mutex.Unlock aufgerufen werden!