8)

a) Was ist der Sinn des Einsatzes von Mutex-Semaphoren ?

b) Worin unterscheiden sich Mutex-Semaphoren von den gewöhnlichen binären Semaphoren ?

Binäres Semaphore	Mutex Semaphore

c) Welche Probleme können im Zusammenhang mit der Nutzung von Mutex-Semaphoren auftreten ? Wie können diese Probleme gelöst werden ?

9) keine Mitschrift; Aufgabentyp: Taskabläufe in Diagramme zeichnen, Auslastungsgrad p des Prozessors (Formel & Rechnung)

10) Gegeben ist das nachstehende VxWorks-Programm. Analysieren Sie dieses und beantworten Sie folgende Fragen.

```
// includes
…
// function prototypes
void taskOne(void);
void taskTwo(void);

// globals
#define ITER 10
SEM_ID semBinary;
int global = 1;

void binary(void) {
        int taskIdOne, taskIdTwo;
        //create semaphore with semaphore available and queue tasks on FIFO basis
        semBinary = semBCreate(SEM_Q_FIFO, SEM_FULL);
        //spawn the two tasks
        taskIdTwo = taskspawn(“t2”, 90, 0x100, 2000, (FUNCPTR)taskTwo, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
        taskIdOne = taskspawn(“t1”, 100, 0x100, 2000, (FUNCPTR)taskOne, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
}
Void taskOne(void) {
int i;
for (i = 0; I < ITER; i++) {
        SemTake(semBinary, WAIT_FOREVER); // wait indefinitely for semaphore
        printf(“I am taskOne and global = &d \n“, --global);
        SemGive(semBinary);
        }
}

void taskTwo(void) {
int i;
for (i = 0; I < ITER; i++) {
        SemTake(semBinary, WAIT_FOREVER); // wait indefinitely for semaphore
        printf(“I am taskTwo and global = &d \n“, ++global);
        SemGive(semBinary);
        }
}
```

a) Printausgabe ? Hinweis: Scheduler ist prioritätsorientiert und preemptive

b) Semaphore wird mit SEM_Q_PRIORITY initialisert. Wie ändert sich die Printausgabe ?

11) Im nachstehenden Programm sind die Semaphorenoperationen jeweils am Anfang und am Ende der Tasks angeordnet.
Ermitteln Sie, ob und in welcher Reihenfolge diese Tasks bei einer Initialisierung nach a), b) und c) ablaufen. SA, SB und SC stellen Counting-Semaphore dar.

Anm.: Der RTOS-Scheduler arbeitet prioritätsorientiert und nach dem „First Come First Serve"-Prinzip, der Kernel ist non-preemptiv. Die Tasks werden in der Reihenfolge TaskA, TaskB, TaskC initialisiert. Alle Tasks weisen die Priorität 90 auf.

```
TaskA {
while(1) {
        SemTake(SA, WFE);
        SemTake(SA, WFE);
        SemTake(SA, WFE);
        …
        // Verarbeitung TaskA
        …
        SemGive(SB);
        }
}
```

```
TaskB {
while(1) {
        SemTake(SB, WFE);
        SemTake(SB, WFE);
        …
        // Verarbeitung TaskB
        …
        SemGive(SC);
        SemGive(SA);
        }
}
```

```
TaskC {
while(1) {
        SemTake(SC, WFE);
        SemTake(SC, WFE);
        SemTake(SC, WFE);
        …
        // Verarbeitung TaskC
        …
        SemGive(SB);
        SemGive(SB);
        }
}
```

a) (nachfolgende Tabellen sind 12 Zeilen tief)

	Werte der Semaphorenvariablen			Reihenfolge der Taskverarbeitung
	SA	SB	SC	
Anfangswerte	1	3	3	

b)

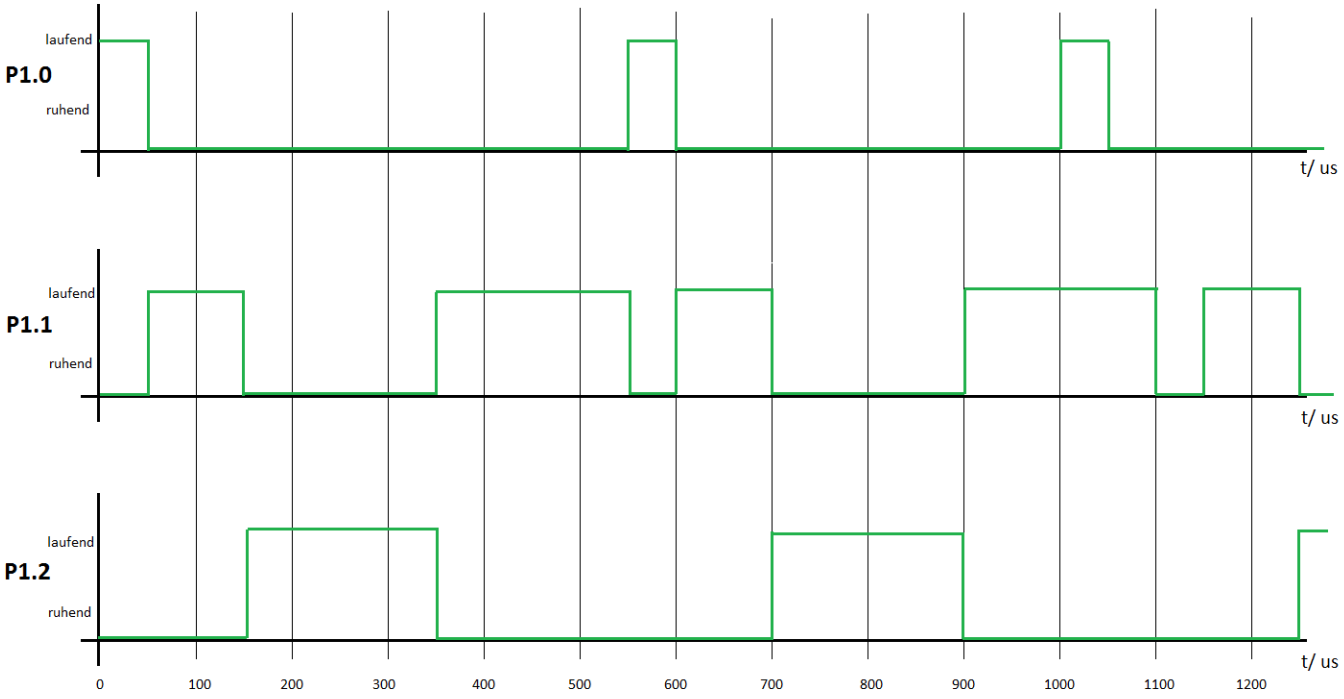
	Werte der Semaphorenvariablen			Reihenfolge der Taskverarbeitung
	SA	SB	SC	
Anfangswerte	1	4	1	

c)

	Werte der Semaphorenvariablen			Reihenfolge der Taskverarbeitung
	SA	SB	SC	
Anfangswerte	3	2	3	

12) Zur Sicherstellung des zyklischen Ablaufs einer Fertigungssteuerung wird das nachfolgende periodische Impulsdiagramm benötigt. Das Impulsdiagramm soll anhand dreier Echtzeit-Tasks auf Basis eines RTOS-Systems erzeugt werden.

Die Signalverläufe werden an den digitalen Portpins P1.0, P1.1, P1.2 mittels der Funktionen SetP1.x_LowHigh() und SetP1.x_HighLow() ausgegeben, wobei x für die Pinnummer 0, 1, 2 steht. Ausserdem steht zur Realisierung der Impulsbreite die Delay-Funktion taskDelay(*Wartezeit_in_us*) zur Verfügung.



a) In welcher Reihenfolge müssen die Tasks ablaufen, damit obiges periodisches Impulsdiagramm realisiert wird ?

Anm.: Der RTOS-Scheduler arbeitet prioritätsorientiert und nach dem „First Come First Serve"-Prinzip, der Kernel ist non-preemptiv. Die Tasks werden in der Reihenfolge TaskA, TaskB, TaskC initialisiert. Alle Tasks weisen die Priorität 90 auf.

b) Fügen Sie entsprechende Semaphorenoperationen, wie SemTake(..), SemGive(..) mit geeigneten Semaphorentypen (Binar, Counting, Mutex) jeweils am Anfang und am Ende der Tasks ein, um den obigen Ablauf sicherzustellen.

```
TaskA {
while(1) {
        …
        SetP1.0_LowHigh();
        taskDelay(50);
        SetP1.0_HighLow();
        …
        }
}
```

```
TaskB {
while(1) {
        …
        SetP1.1_LowHigh();
        taskDelay(100);
        SetP1.1_HighLow();
        …
        }
}
```

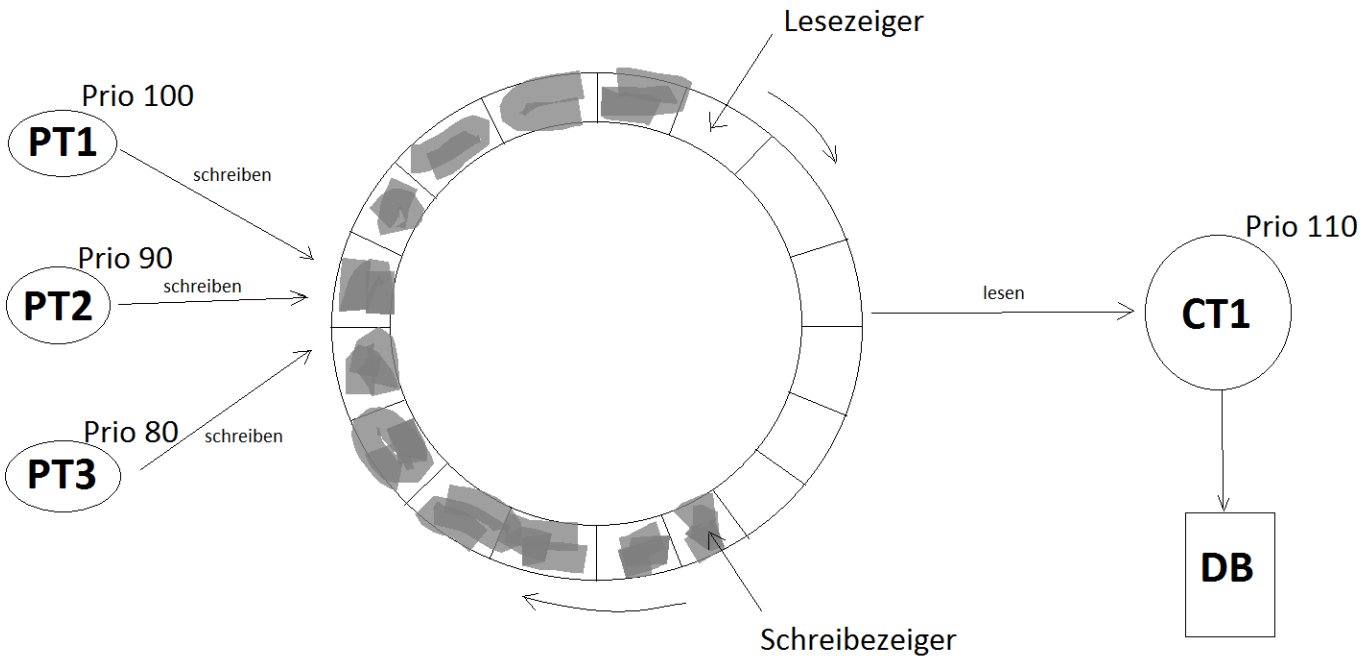
```
TaskC {
while(1) {
        …
        SetP1.2_LowHigh();
        taskDelay(200);
        SetP1.2_HighLow();
        …
        }
}
```

c) Ermitteln Sie die Anfangswerte für die Semaphoren SA, SB und SC.

13) Welche Abhängigkeiten bestehen zwischen der Interrupt-Anforderung (IRQ), der Interrupt-Serviceroutine (ISR) und der Interrupt-Servicetask (IST) ?

14) Ein Ringpuffer ist ein nach dem FIFO-Prinzip organisierter Speicher zur Pufferung von Daten. Im gegebenen Fall sollen drei Protokolliertask (PT-Producer Tasks) und ein Archivierungstask (CT-Consumer Task) auf diesen Ringpuffer zugreifen. Der Ringpuffer befindet sich im globalen linear adressierten Speicherbereich.

Die Daten fallen kontinuierlich an und werden von den drei Producer-Tasks kontinuierlich in den Ringpuffer geschrieben und ggf. zeitversetzt schritthaltend von einem Consumer-Task weiterverarbeitet, d.h. in eine Datenbank (DB) abgespeichert. Dem Anwendungssystem liegt ein preemptiver Scheduler zugrunde. Die Prioritäten der Task sind dem untenstehenden Bild zu entnehmen.



a) Wie groß ist der Ringpufferspeicher mindestens zu wählen ? (Abhängigkeiten ?)

b) Welche Mindestanforderungen sind an die Ringpufferverwaltung zu stellen ? Welche Synchronisationsaufgaben existieren ?

15) Gegeben sei eine Tankstelle, bestehend aus zwei Zapfsäulen mit jeweils 2x3 Zapfpistolen für Benzin, Super-Benzin und Diesel. Es existieren vier Tankplätze, so dass gleichzeitig maximal vier Fahrzeuge betankt werden können. Ausserdem existiert noch eine zentrale Bezahlstation, an der alle betankten Fahrzeuge nacheinander vorbeifahren und bezahlen können.

a) Welche Synchronisationsaufgaben sind zu bewältigen, um einen ordnungsgemäßen Betrieb sicherzustellen ?