Übung 2

Aufgabe 1 *Semaphore in C/Real-Time POSIX*

Programmieren Sie eine Lösung des Erzeuger-/Verbraucher-Problems mit Hilfe der Semaphore-Schnittstelle in C/Real-Time POSIX (vgl. Vorlesung S. 3-17 – 3-20).

Aufgabe 2 *Semaphore in Ada*

Programmieren Sie eine Lösung des Problems der gemeinsamen Ressourcen (vgl. Vorlesung S. 3-22 – 3-25) in Ada mit Hilfe des auf S. 3-17 deklarierten Semaphor-Packages.

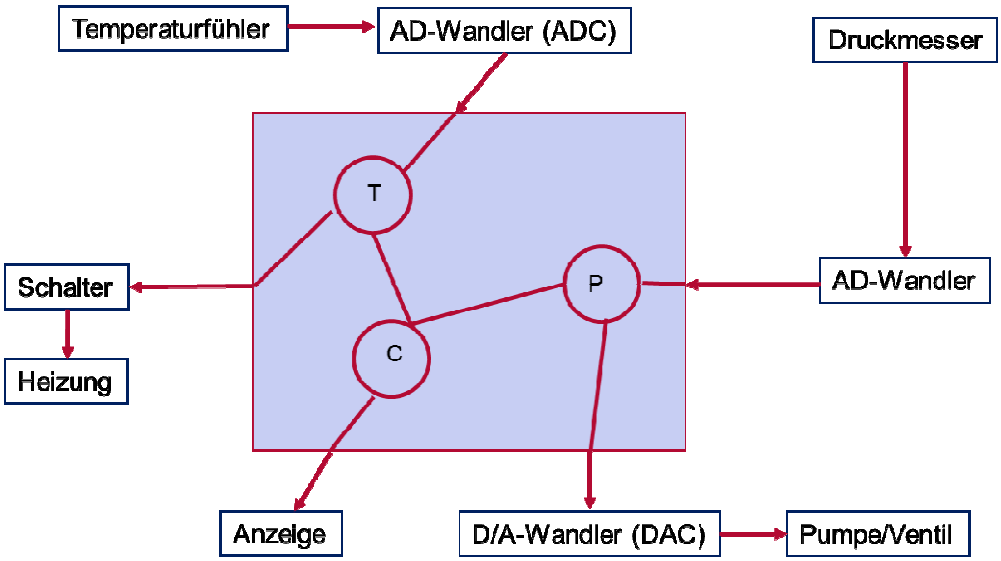
Aufgabe 3 *Geschützte Objekte in Ada: Implementierung eines Semaphore-Packages*

Implementieren Sie das auf S. 3-17 deklarierte Semaphor-Package von Ada

Hinweis: Implementieren Sie den Typ *Semaphore* im Privaten Teil der Package-Deklaration als geschütztes Objekt mit speziellen *Wait*- und *Signal*-Operationen als geschützten Operationen. Die öffentlichen *Wait*- und *Signal*-Operationen können dann mit Hilfe der geschützten Operationen implementiert werden.

Aufgabe 4 *Prozesssteuerung in Ada mit Hilfe von geschützten Objekten*

Das "einfache eingebettete System" aus der Vorlesung (S. 2-41 ff) kann durch eine dritte Task ergänzt werden, die für das Entgegennehmen der gemessenen Druck und Temperaturwerte sowie deren Darstellung an der Konsole zuständig ist.



Die Kommunikation zwischen den Messprozessen T und P sowie dem Display-Prozess C kann über ein geschütztes Objekt *Console_Data* erfolgen. Die Tasks T und P schreiben ihre gemessenen Werte in *Console_Data*, die Task T liest immer dann die Werte aus *Console_Data*, wenn dort neue Werte geschrieben worden sind.

Skizzieren Sie eine Implementierung des Systems mit drei Tasks in Ada!

Hinweise:

Die Prozedur *Controller* mit den Tasks *Temp_Controller* und *Pressure_Controller* der nebenläufigen ADA-Lösung kann unverändert bleiben. Es ist lediglich das Package IO (S. 2-43) entsprechend zu implementieren.

```
with Data_Types; use Data_Types;
package IO is
  procedure Read(TR : out Temp_Reading); -- from ADC
  procedure Read(PR : out Pressure_Reading);
  procedure Write(HS : Heater_Setting);-- to switch
  procedure Write(PS : Pressure_Setting); -- to DAC
  procedure Write(TR : Temp_Reading);  -- to screen
  procedure Write(PR : Pressure_Reading);-- to screen
end IO;
```

Die Lese-/und Schreib-Methoden für die Sensoren (Druck und Temperatur) und Aktoren (Heizung und Ventil/Pumpe) können nach wie vor als gegeben vorausgesetzt werden.

Die letzten beiden Schreib-Methoden (auf den Bildschirm) müssen ersetzt werden durch ein Schreiben in das geschützte Objekt *Console_Data*.

Im Implementierungsteil (body) des Packages *IO* sind also das geschützte Objekt *Console_Data*, die letzten beiden Write-Methoden (schreiben nicht auf den Screen, sondern in *Console_Data*) sowie die Task *Console* zu implementieren, die *Console_Data* liest und die Werte auf dem Bildschirm ausgibt. Dabei ist zu realisieren, dass die Task *Console* nur Daten aus *Console_Data* liest, die neu geschrieben wurden, d.h. die er nicht schon gelesen hat. Zugriffe auf *Console_Data* sollten sich grundsätzlich gegenseitig ausschließen.

Aufgabe 5 *Zufahrtskontrolle zu einem Parkplatz*

Lösen Sie das Parkplatzproblem (Vgl. Übung 1, Aufgabe 5) mit Hilfe dreier nebenläufiger Tasks: eine steuert die Einfahrtsschranke, eine steuert die Ausfahrtsschranke und eine steuert das Signal.

Betrachten Sie die Anzahl der Fahrzeuge auf dem Parkplatz als gemeinsame Variable.

Beschreiben Sie die Synchronisationsbedingungen, die von den drei Prozessen eingehalten werden müssen.

Skizzieren Sie Lösungen des Synchronisationsproblems

(a) mit Hilfe von Mutexen und Bedingungsvariablen in C/Real-Time POSIX und

(b) mit Hilfe von geschützten Objekten in Ada.