Betrieb eines Modus

Für die Qualität ihrer Lösung ist entscheidend, daß der Ablauf zu 100% kompatibel mit der Spezifikation des Standardprotokolls ist.

Sie dürfen ein Modul nicht starten, wenn es kein Ready=1 liefert.

Die Handshakequittung ist Acknowledge. Das funktioniert auch mit Busy, ist dann aber nicht spezifikationstreu. Wenn Sie z.B. einen Nullauftrag geben (Oder=0), funktioniert es nicht mehr !

Erste Aufgabe, Starten eines Moduls :

```
from twinLib2021 import *
sps = PlcModule()

nix = input("zum Start ENTER drücken")

sps.StartBand()

ready = sps.ReadOPCTag('module1','Ready')
busy = sps.ReadOPCTag('module1','Busy')

if ready == 1 and busy == 0 :

    sps.WriteOPCTag('module1','Order',1)
    sps.WriteOPCTag('module1','Start',1)
    ack = sps.ReadOPCTag('module1','Acknowledge')
    while ack == 0:
        ack = sps.ReadOPCTag('module1','Acknowledge')
    sps.WriteOPCTag('module1','Start',0)

else :

    print("Modul nicht funktionsbereit !")
```

Hier wird peinlich genau das Protokoll eingehalten.
Wenn man sicher ist, daß das ein einmaliger Test ist und sonst niemand an der Anlage arbeitet, könnte man natürlich auf die Prüfung von Busy verzichten : Das ist keine Änderung am, Handshake !

Zweite Aufgabe : Produktvariante abwechseln

a) mit Polling-Struktur

```
from twinLib2021 import *
sps = PlcModule()

nix = input("zum Start ENTER drücken")

sps.StartBand()
anzahl=0

ready = sps.ReadOPCTag('module1','Ready')
busy = sps.ReadOPCTag('module1','Busy')

if ready == 1 and busy == 0 :

    while anzahl < 10 :

        sps.WriteOPCTag('module1','Order',1)
        sps.WriteOPCTag('module1','Start',1)
        ack = sps.ReadOPCTag('module1','Acknowledge')
        while ack == 0:
            ack = sps.ReadOPCTag('module1','Acknowledge')
        sps.WriteOPCTag('module1','Start',0)
        busy = sps.ReadOPCTag('module1','Busy')
        while busy == 1:
            busy = sps.ReadOPCTag('module1','Busy')

        sps.WriteOPCTag('module1','Order',2)
        sps.WriteOPCTag('module1','Start',1)
        ack = sps.ReadOPCTag('module1','Acknowledge')
        while ack == 0:
            ack = sps.ReadOPCTag('module1','Acknowledge')
        sps.WriteOPCTag('module1','Start',0)
        busy = sps.ReadOPCTag('module1','Busy')
        while busy == 1:
            busy = sps.ReadOPCTag('module1','Busy')

        anzahl = anzahl+2
'
```

Die Wartevorgänge in den polling-Schleifen steuern den zeitlichen Ablauf !

b) mit state-machine :

```
from twinLib2021 import *
sps = PlcModule()

nix = input("zum Start ENTER drücken")

sps.StartBand()
anzahl=0
state=1

ready = sps.ReadOPCTag('module1','Ready')
busy = sps.ReadOPCTag('module1','Busy')

if ready == 1 and busy == 0 :

    while anzahl < 10 :

        if state==1:
            busy = sps.ReadOPCTag('module1','Busy')
            if busy == 0:
                sps.WriteOPCTag('module1','Order',1)
                sps.WriteOPCTag('module1','Start',1)
                state=2

        if state==2 :
            ack = sps.ReadOPCTag('module1','Acknowledge')
            if ack == 1:
                sps.WriteOPCTag('module1','Start',0)
                state=3
                anzahl=anzahl+1

        if state==3 :
            busy = sps.ReadOPCTag('module1','Busy')
            if busy == 0:
                sps.WriteOPCTag('module1','Order',2)
                sps.WriteOPCTag('module1','Start',1)
                state=4

        if state==4 :
            ack = sps.ReadOPCTag('module1','Acknowledge')
            if ack == 1:
                sps.WriteOPCTag('module1','Start',0)
                state=1
                anzahl=anzahl+1
```

Das kann man natürlich auch knapper schreiben. Hier ist absichtlich die state-machine deutlich hervorgehoben. Der Sprung von state zu state steuert den zeitlichen Ablauf !