serielle Kopplung

Praxis

Auf den folgenden Seiten finden sie zwei Lösungsvorschläge.

Einmal mit Warteschleifen (polling), einmal als stark übertrieben strukturierte state-machine. Ich habe das so ausführlich geschrieben, um die state-Struktur hervorzuheben.

Die Lösungen, die ich von ihnen bekommen habe, realisieren zum größten Teil den Handshake nicht richtig.

twintest_polling.py - C:/Users/r_user/Desktop/digitalTwin/twintest_polling.py (3....

File Edit Format Run Options Window Help

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

los = input("zum Start ENTER drücken")

sps.StartBand()


while True :

    sps.WriteOPCTag('module1','Order',1)
    sps.WriteOPCTag('module1','Start',1)

    ack = sps.ReadOPCTag('module1','Acknowledge')
    while ack == 0:      #warte auf handshakequittung
        ack = sps.ReadOPCTag('module1','Acknowledge')

    sps.WriteOPCTag('module1','Start',0)

    busy = sps.ReadOPCTag('module1','Busy')
    while busy == 1:     #warte, bis modul fertig
        busy = sps.ReadOPCTag('module1','Busy')

    #-------------------------------------------------

    sps.WriteOPCTag('module2','Order',1)
    sps.WriteOPCTag('module2','Start',1)

    ack = sps.ReadOPCTag('module2','Acknowledge')
    while ack == 0:      #warte auf handshakequittung
        ack = sps.ReadOPCTag('module2','Acknowledge')

    sps.WriteOPCTag('module2','Start',0)

    busy = sps.ReadOPCTag('module2','Busy')
    while busy == 1:     #warte, bis modul fertig
        busy = sps.ReadOPCTag('module2','Busy')

    #-------------------------------------------------

    sps.WriteOPCTag('module3','Order',1)
    sps.WriteOPCTag('module3','Start',1)

    ack = sps.ReadOPCTag('module3','Acknowledge')
    while ack == 0:      #warte auf handshakequittung
        ack = sps.ReadOPCTag('module3','Acknowledge')

    sps.WriteOPCTag('module3','Start',0)

    busy = sps.ReadOPCTag('module3','Busy')
    while busy == 1:     #warte, bis modul fertig
        busy = sps.ReadOPCTag('module3','Busy')
```

Ln: 16 Col: 53

seriell_statemachine.py - C:\Users\r_user\Desktop\digitalTwin\seriell_statemachine.py (3.8.3)

File Edit Format Run Options Window Help

```
from twinLib2021 import *
sps = PlcModule()
state=1

start = input("zum Start ENTER drücken")
sps.StartBand()

while True :

#-----------------------------------------------------------------

    if state==1:
        sps.WriteOPCTag('module1','Order',1)        #modul 1 starten
        sps.WriteOPCTag('module1','Start',1)
        state=2                                     #der nächste zustand wird aktiviert

    if state==2:
        ack = sps.ReadOPCTag('module1','Acknowledge')
        if ack==1:                                  #hier wird abgefragt, ob die quittung eingetroffen ist
            sps.WriteOPCTag('module1','Start',0)    #wenn ja, wird start zurückgenommen : modul 1 läuf jetzt
            state=3                                 #der nächste zustand wird aktiviert

    if state==3:
        busy = sps.ReadOPCTag('module1','Busy')     #in diesem schritt wird geprüft, ob modul 1 fertig ist
        if busy==0:
            state=10                                #weiter mit nächstem modul

#-----------------------------------------------------------------

    if state==10:
        sps.WriteOPCTag('module2','Order',1)        #modul 2 starten
        sps.WriteOPCTag('module2','Start',1)
        state=11                                    #der nächste zustand wird aktiviert

    if state==11:
        ack = sps.ReadOPCTag('module2','Acknowledge')
        if ack==1:                                  #hier wird abgefragt, ob die quittung eingetroffen ist
            sps.WriteOPCTag('module2','Start',0)    #wenn ja, wird start zurückgenommen : modul 2 läuft jetzt
            state=12                                #der nächste zustand wird aktiviert

    if state==12:
        busy = sps.ReadOPCTag('module2','Busy')     #in diesem schritt  wird geprüft, ob modul 2 fertig ist
        if busy==0:
            state=20                                #weiter mit nächstem modul

#-----------------------------------------------------------------

    if state==20:
        sps.WriteOPCTag('module3','Order',1)        #modul 3 starten
        sps.WriteOPCTag('module3','Start',1)
        state=21                                    #der nächste zustand wird aktiviert

    if state==21:
        ack = sps.ReadOPCTag('module3','Acknowledge')
        if ack==1:                                  #hier wird abgefragt, ob die quittung eingetroffen ist
            sps.WriteOPCTag('module3','Start',0)    #wenn ja, wird start zurückgenommen : modul 3 läuft jetzt
            state=22                                #der nächste zustand wird aktiviert

    if state==22:
        busy = sps.ReadOPCTag('module3','Busy')     #in diesem schritt  wird geprüft, ob modul 3 fertig ist
        if busy==0:
            state=1                                 #weiter mit nächstem modul

#-----------------------------------------------------------------
```

Ln: 60 Col: 71

Dazu optional noch der Redundanzbetrieb :

```
from twinLib2021 import*
twin= PlcModule()
a= input("drücke Enter zum Starten")
twin.StartBand()

ready = twin.ReadOPCTag("module1","Ready")
busy = twin.ReadOPCTag("module1","Busy")
counter=0
redundanz="3a"

if ready==1 and busy==0:
   while counter<5:
      #Modul 1 Ablauf
      busy = twin.ReadOPCTag("module1", "Busy")
      ready = twin.ReadOPCTag("module1","Ready")
      while busy == 1 or ready==0:              #wenn modul 1 betriebsbereit
         ready = twin.ReadOPCTag("module1", "Ready")
         busy = twin.ReadOPCTag("module1", "Busy")
      twin.WriteOPCTag("module1","Order",1)
      twin.WriteOPCTag("module1","Start",1)
      ack = twin.ReadOPCTag("module1", "Acknowledge")
      while ack == 0:
         ack = twin.ReadOPCTag("module1", "Acknowledge")
      twin.WriteOPCTag("module1", "Start", 0)

      #Modul 2 Ablauf
      busy = twin.ReadOPCTag("module1", "Busy")
      ready = twin.ReadOPCTag("module2","Ready")
      while busy == 1 or ready==0:              #polling, bis 1 fertig
         ready = twin.ReadOPCTag("module2", "Ready")
         busy = twin.ReadOPCTag("module1", "Busy")
      twin.WriteOPCTag("module2", "Order", 1)
      twin.WriteOPCTag("module2", "Start", 1)
      ack = twin.ReadOPCTag("module2", "Acknowledge")
      while ack == 0:
         ack = twin.ReadOPCTag("module2", "Acknowledge")
      twin.WriteOPCTag("module2", "Start", 0)

      #Modul 3 Ablauf
      busy = twin.ReadOPCTag("module2", "Busy")
      ready = twin.ReadOPCTag("module3","Ready")
      while busy == 1 or ready==0:              #polling, bis 2 fertig
         ready = twin.ReadOPCTag("module3", "Ready")
         busy = twin.ReadOPCTag("module2", "Busy")
      #
```

```
        #
        if redundanz == "3b" :
            twin.WriteOPCTag("module3b", "Order", 1)
            twin.WriteOPCTag("module3b", "Start", 1)
            ack = twin.ReadOPCTag("module3b", "Acknowledge")
            while ack == 0:
                ack = twin.ReadOPCTag("module3b", "Acknowledge")
            twin.WriteOPCTag("module3b", "Start", 0)
            busyb = twin.ReadOPCTag("module3b", "Busy")
            while busyb == 1:                       #polling, bis 3 fertig
                busyb = twin.ReadOPCTag("module3b", "Busy")
            #
            twin.WriteOPCTag("module3", "Order", 6)
            twin.WriteOPCTag("module3", "Start", 1)
            ack = twin.ReadOPCTag("module3b", "Acknowledge")
            while ack == 0:
                ack = twin.ReadOPCTag("module3", "Acknowledge")
            twin.WriteOPCTag("module3", "Start", 0)
            #
            twin.WriteOPCTag("module3", "Order", 8)
            twin.WriteOPCTag("module3", "Start", 1)
            ack = twin.ReadOPCTag("module3", "Acknowledge")
            while ack == 0:
                ack = twin.ReadOPCTag("module3", "Acknowledge")
            twin.WriteOPCTag("module3", "Start", 0)
            redundanz = "3a"
        else:
            twin.WriteOPCTag("module3", "Order", 1)
            twin.WriteOPCTag("module3", "Start", 1)
            ack = twin.ReadOPCTag("module3", "Acknowledge")
            while ack == 0:
                ack = twin.ReadOPCTag("module3", "Acknowledge")
            twin.WriteOPCTag("module3", "Start", 0)
            redundanz = "3b"


        #warte, bis 3a oder 3b fertig, dann wieder von vorne

        busy = twin.ReadOPCTag("module3", "Busy")
        busyb = twin.ReadOPCTag("module3b", "Busy")
        while busy == 1 or busyb == 1:                  #polling, bis 3 fertig
            busy = twin.ReadOPCTag("module3", "Busy")
            busyb = twin.ReadOPCTag("module3b", "Busy")

        #Zähler
        counter=counter+1
    else:
        print("Anlage noch nicht einsatzbereit!")
```