Technikerschule - Fachschule für
Maschinenbau-, Metallbau-, Informatik- und Elektrotechnik
der Landeshauptstadt München

Technikerprüfung 2014/15

Automatisierungstechnik

Zeit : 150 Minuten

Klasse :	Name : Muster

	Punkte:	Note :	Unterschriften:
Erstkorrektur			
Zweitkorrektur			

Anlagenbeschreibung : Behalten Sie dieses Blatt auch für den Teil mit Unterlagen !

Wir betrachten eine Anlage zur Montage von Armbanduhren.
Die Uhren werden in **Losgröße 1** hergestellt, die **Minimalcharge ist 20**.
Der Kunde kann sich seine Uhr im Onlineshop konfigurieren, der Auftrag wird in der SQL-Datenbank im ERP-System unter fortlaufender Bestellnummer gespeichert.

Die Uhr besteht aus einem **Deckel**, einem **Werk** und einem **Gehäuse**, das als Kunststoff-Spritzgußteil gefertigt wird. Jedes Bauteil kann aus mehreren Varianten gewählt werden.

Das Granulat für den Spritzguß wird in einem Nachbarbetrieb in einem Recyclingverfahren gewonnen und nach Bedarf mit einem Flugförderer (wie ein großer Staubsauger) über 50m in die eigene Werkshalle gefördert.

ERP-Ebene :

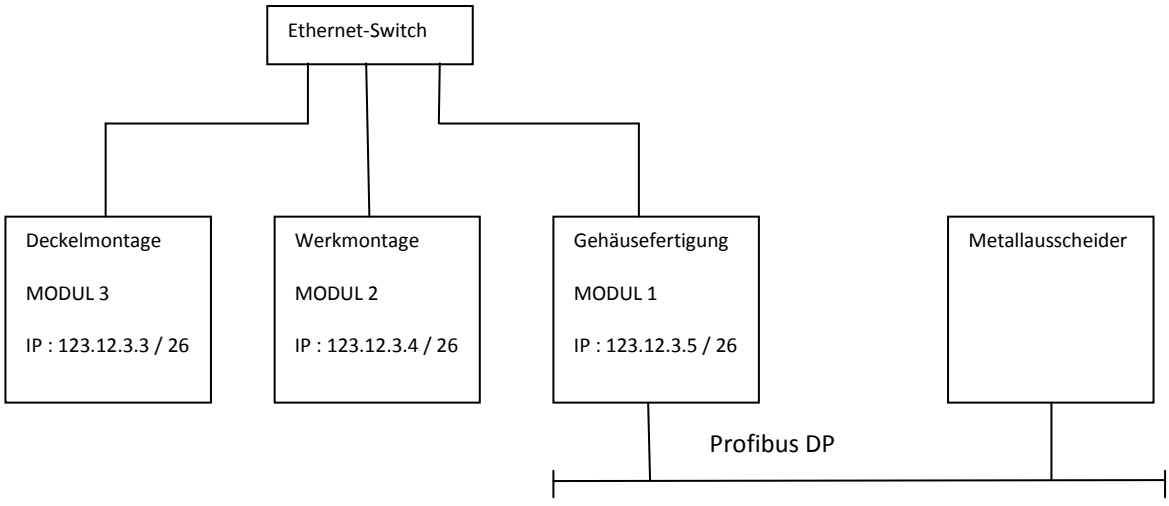
Die Anlagen-IT besteht aus einer Windows-Domäne mit einem Win2008 Domänencontroller.
Das ERP-System läuft ebenfalls auf Win 2008Server.
Die Kommunikation zwischen MES- und ERP-Ebene geschieht über Webservices.

MES-Ebene :

Das MES-System ist ein 2008Server mit einem in VB geschriebenen MES-Kern.
Zur Kommunikation mit ERP und Prozessebene steht eine Klassenbibliothek zur Verfügung.
Die Prozessebene (Modul 1 bis 3) wird darin über ISO-on-TCP angesprochen.

Prozessebene :

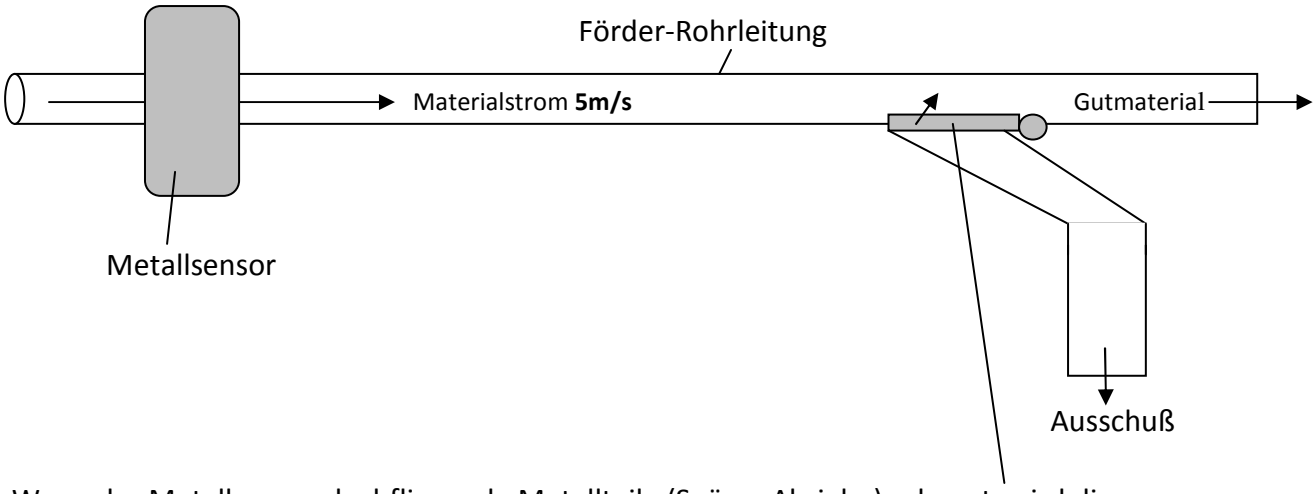
Jedes Montagemodul (Deckel, Werk, Gehäuse) wird von einer Simatic S7 315-2 dp/pn gesteuert. Die SPS sind über Ethernet-Switches mit dem Anlagennetz verbunden.
Der Transport von einem Modul zum Nächsten ist Teil der Modulfunktion.
Zur Absicherung der Spritzgußeinheit vor Metallspänen o.Ä. ist am Flugförderer ein Metallausscheider montiert, der über Profibus DP angesteuert wird. Auf die Anlagenfunktion aus MES-Sicht hat dieser keinen Einfluß.



Teil 1 , ohne Unterlagen

Klasse : Name :

Das Bild zeigt den mechanischen Aufbau eines Metallausscheiders :



Wenn der Metallsensor duchfliegende Metallteile (Späne, Abrieb..) erkennt, wird die Separatorenklappe kurz geöffnet. Die Metallteile und etwas Granulat werden in den Ausschuß abgelenkt. Es dürfen keinesfalls Metallteile in das Gutmaterial gelangen, weil sonst der Spritzguß-maschine Zerstörung droht.

Der Abstand zwischen Sensor und Klappe beträgt 1 Meter.

1. Aufgabe :

Kreuzen Sie die nachfolgenen Kommunikationssysteme an, die Ihrer Meinung nach für diese Applikation brauchbar sind :

O	Profinet I/O	**4 Punkte (½ pro Auswahl)**
O	Profinet CBA	
X	Profinet IRT	
X	Profibus FMS	
X	Profibus DP	
O	UDP/IP auf Ethernet	
O	TCP/IP auf Ethernet	
O	Webservices XML	

2. Aufgabe :

Begründen Sie ihre Auswahl. Welches gemeinsame Kriterium macht die als „brauchbar“ gewählten Systeme aus :

……

Determinismus ! **(2 Punkte)**

……

3. Aufgabe :

Beschreiben Sie nun bei jedem brauchbaren System, wie die relevante Funktion im Detail technisch realisiert ist :

……

Profibus : deterministischer Layer2 durch Token-Verfahren

(2 Punkte)

Profinet IRT : synchrone Echtzeitkommunikation in speziellem Zeitschlitz

Realisiert durch synchrone Echtzeituhren in den Teilnehmern

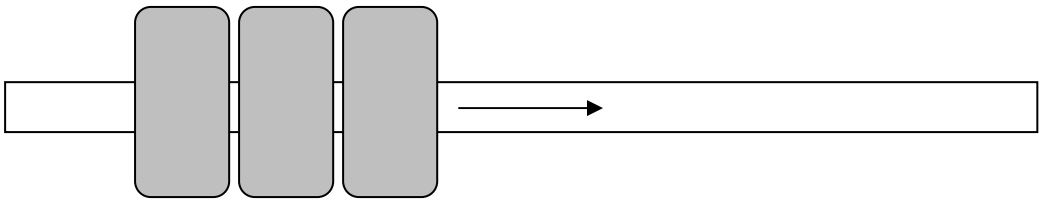
(3 Punkte)

……

……

In einem Umfeld, in dem sowohl der Preis des Granulats sehr hoch, als auch der durch Metallpartikel im Granulat mögliche Schaden an der Spritzgußmaschine sehr kostspielig ist, wäre eine statisch redundante Ausführung des Metallsensors möglich.

Im Nachfolgenden wird daher mit 3 Sensoren an der Förderleitung gearbeitet.



4. Aufgabe :

Was bedeutet „statische Redundanz“ ?

Alle Redundanzkomponenten laufen im Betrieb **(2 Punkte)**

5. Aufgabe :

Versuchsweise sollen die Sensoren im TMR-Betrieb laufen. Was bedeutet das ?

……

Drei Komponenten arbeiten. Bei Abweichung einer Komponente wird diese als

Fehlerhaft betrachtet und abgeschalten. **(2 Punkte)**

……

……

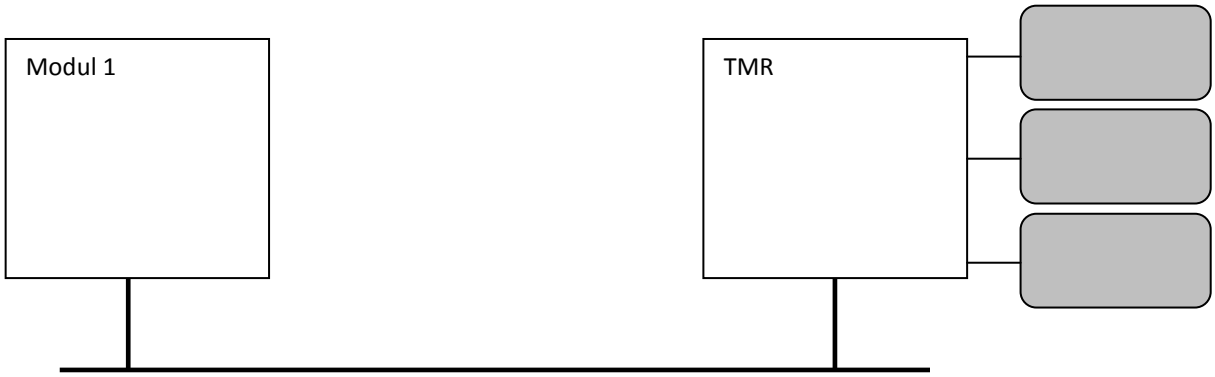
Teil 2, mit Unterlagen

Alle nachfolgenden Aufgaben bearbeiten Sie bitte auf dem karierten Lösungsblatt !

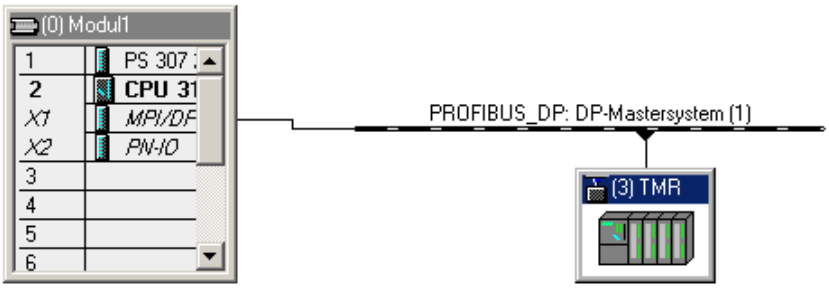
Der im Teil ohne Unterlagen schon angesprochene TMR-Betrieb der drei Metallsensoren wird durch eine SPS "**TMR**" ausgeführt. Ein FB rechnet die TMR-Funktion. Als Ausgang dient der Merker M200.0 , dieser wird bei erkanntem Metallteil = 1.

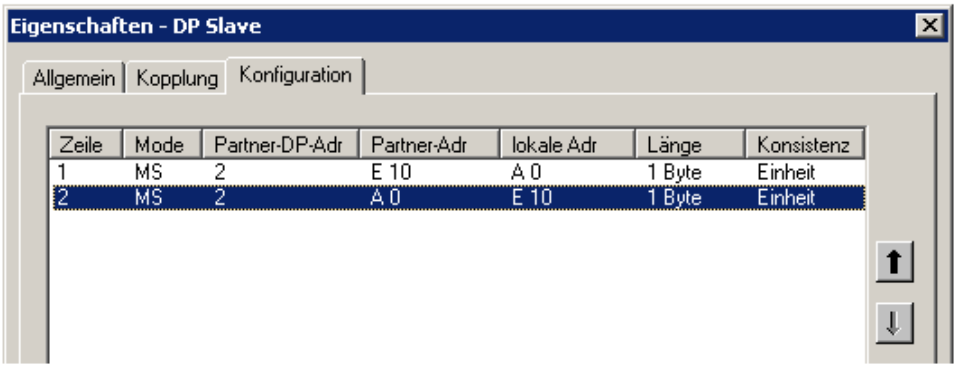
Dieses Signal wird mit Profibus DP an die SPS "**Modul1**" übertragen, welche die Spritzgußmaschine und den Antrieb der Separatorenklappe steuert :

Blockbild :



Hier sehen Sie die Konfiguration der SPS "Modul1" (Master) :





1. Aufgabe :

Zeichnen Sie die aus der Hardwarekonfiguration ersichtlichen Ein- / Ausgabezellen in einem Blockbild wie oben in diejenigen Geräte ein, in denen sie sich physikalisch befinden. Kennzeichnen Sie auch, welche Zellen logisch zu welchem Gerät gehören.

(2 Punkte Zellen logisch am Master)

(2 Punkte Zellen physikalisch im Slave)

2. Aufgabe :

Geben Sie zu TMR und Modul1 die nötigen Programmteile an, um den Inhalt von M200.0 aus TMR in den Merker 100.0 in Modul1 zu transportieren (AWL bitte) .

Programm TMR :

U M200.0 **(2 Punkte)**
= A0.0

Programm Modul1 :

U E10.0 **(2 Punkte)**
= M 100.0

Zwischen SPS und MES-System wird zur Kommunikation ein Handshakeverfahren eingesetzt. Die **SPS** setzt bei Anlauf das Signal **READY .**

Will **MES** die SPS starten, wird zunächst mit einem Signal **TEST** in der SPS ein Selbsttest ausgelöst. (TEST darf nur erfolgen, wenn READY = 1)

Die **SPS** setzt daraufhin **READY** = 0. (Dies dient als Quittung für TEST)

(Mit dem internen Signal OK wird in der SPS der erfolgreiche Selbsttest gemeldet. Daraufhin setzt die SPS das Signal READY wieder 1. Ist der Selbsttest nicht erfolgreich, bleibt READY = 0, die SPS muß dann manuell rückgesetzt werden)

Nach wieder erfolgtem READY = 1 löst **MES** die SPS-Mechankfunktion mit **START** aus.

Im Sinne eines Funktionshandshakes wird **READY** von der **SPS** wieder = 0 gesetzt.

(Das relevante Funktionsende wird durch das interne Signal END in der SPS gemeldet, damit wird READY wieder 1 gesetzt)

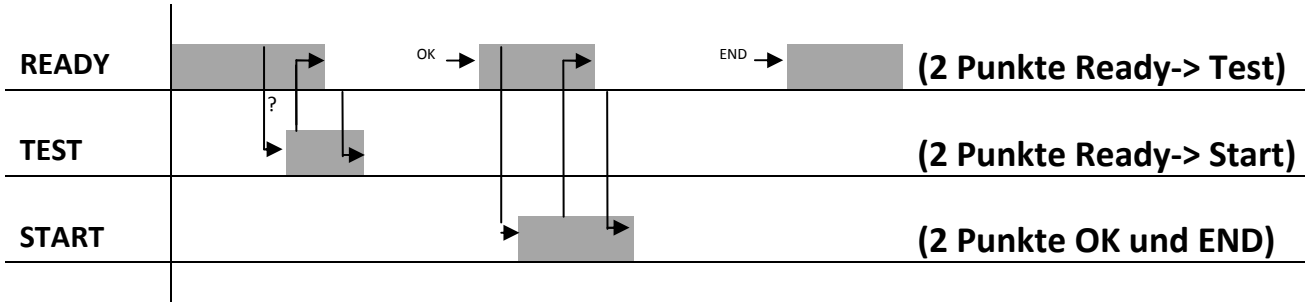
3. Aufgabe :

Erklären Sie kurz, was man unter „relevantem Funktionsende“ versteht

Der Zeitpunkt in einer Modulfunktion, ab dem das Modul die Funktion anderer Module nicht mehr behindert **(2 Punkte)**

4.Aufgabe :

Zeichnen Sie ein Timing-Diagramm, das alle BUS-Kommunikationssignale für eine SPS enthält



(2 Punkte Ready-> Test)

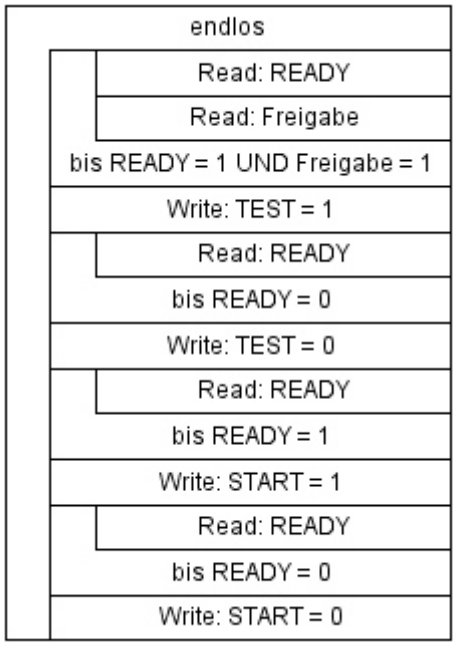
(2 Punkte Ready-> Start)

(2 Punkte OK und END)

SUMME : 6 Punkte

5. Aufgabe :

Zeichnen Sie einen graphischen Entwurf für den Programmteil im MES-System, der den Handshake mit einer SPS ausführt. Auslöser soll eine Benutzereingabe “Freigabe“ sein. (Die Auswahl eines geeigneten graphischen Verfahrens ist Teil der Aufgabe)



Ready, Freigabe in Polling -> 2 Punkte

Test = 1 -> 1 Punkt

Ready in Polling -> 1 Punkt

Damit Test = 0 -> 1 Punkt

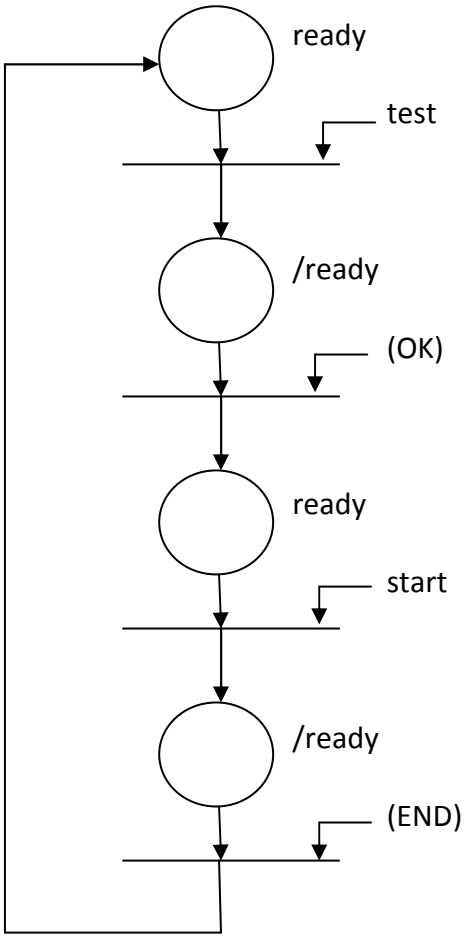
Wieder Ready-Polling -> 1 Punkt

Start = 1, Polling Ready, Start = 0 -> 2 Punkte

SUMME : 8 Punkte !

6. Aufgabe :

Zeichnen Sie einen graphischen Entwurf für den Programmteil auf SPS-Seite des Handshakes . (Die Auswahl eines geeigneten graphischen Verfahrens ist Teil der Aufgabe)



Anfang -> 1 Punkt

Start mit test -> 1 Punkt

/ready -> 1 Punkt

intern : ok

ready -> 1 Punkt

start -> 1 Punkt

end -> 1 Punkt

SUMME : 6 Punkte !

Im Weiteren wird die Kommunikation zwischen SPS und MES mit einer Klassenbibliothek TP_2015.dll erleichtert, die hierfür folgende Klassen bereitstellt :

Klasse : MODUL_CONTROL

Name :	MODUL_CONTROL
Attribute :	Modul_IP (String) Variante (Integer) Status(String)
Methoden :	Lies_Status() Start Modul()

Die Module müssen mit ihren IP-Adressen (xxx . xxx . xxx . xxx) parametriert werden.
Im Attribut **Variante** wird dem Modul mitgeteilt, welche Fertigungsvariante durchzuführen ist. Hier ist der Nullauftrag möglich, der keine mechanische Funktion auslöst.
Die Methode **Start_Modul()** führt den Start-Handshake mit der SPS aus.
Die Methode **Lies_Status()** liest aus der SPS den aktuellen Betriebszustand und gibt ihn als String im Attribut **Status** aus. Möglich sind hier : RUNNING und OK

Die Abfrage der Varianteninformation erfolgt aus der Datenbank des ERP-Systems mit der

Klasse : ERP_CONNECT

Name :	ERP_CONNECT
Attribute :	ID(String) Teil (String) Variante(Integer) Exists(Byte)
Methoden :	Lies_teil()

Es muß die **ID** des Produkts (Bestellnummer, beginnend mit 1) angegeben werden, dazu welches **Teil** (“Gehäuse“, “Werk“ oder „“Deckel“). Dann kann mit Lies_teil() über Webservices eine Abfrage der Variante aus der Datenbank erfolgen. Die Variante steht dann im Attribut **Variante**.
Wenn ein Auftrag für die angegebene ID nicht existiert, wird Exists = 0, ansonsten = 1.

7. Aufgabe :

Zeichnen Sie einen Workplan für den Fertigungsanlauf der Anlage.

	Schritt 1	Schritt 2	Schritt 3	
Modul 1	**G1**	**G2**	**G3**	**1 Punkt**
Modul 2	**0**	**W1**	**W2**	**1 Punkt**
Modul 3	**0**	**0**	**D1**	**1 Punkt**

Summe : 3 Punkte

8. Aufgabe :

Geben Sie ein Visual Basic – Programm (oder C#) an, das den Fertigungsanlauf aus Aufgabe 7 realisiert, falls ausreichend Bestellungen eingegangen sind. Die Bestellungen beginnen mit der ID 1, der Ablauf der Programmlogik durch einen Button-Click.

```
Imports   TP_2015.dll                                          Library -> 1 Punkt
Public Class Form 1

    Public M1 as new MODUL_CONTROL                             Referenzen -> 2 Punkte
    Public M2 as new MODUL_CONTROL
    Public M3 as new MODUL_CONTROL
    Public ERP as new ERP_CONNECT

    M1.Modul_IP = "123.12.3.3"                                 IP -> 2 Punkte
    M2.Modul_IP = "123.12.3.4"
    M3.Modul_IP = "123.12.3.5"

    Private Sub Button1_Click() handles Button1_Click          Button -> 1 Punkt

        ERP.ID = "20"                  '----------------------Test ob 20 Bestellungen
        ERP.Teil = "Werk"
        Do
            ERP.Lies_teil()
        loop while ERP.Exists = 0                              Test ob 20 -> 2 Punkte

        ERP.ID = "1"                   '-----------------------Schritt 1
        ERP.Teil = "Gehäuse"
        ERP.Lies_teil()
        M1.Variante = ERP.Variante                             M1 lesen -> 1 Punkt

        M2.Variante = 0
        M3.Variante = 0                                        Nullaufträge -> 1 Punkt
        M1.Start_Modul()
        M2.Start_Modul()
        M3.Start_Modul()
        Do
            M1.Lies_Status()                                   Polling fragen-> 1 Punkt
            M2.Lies_Status()
            M3.Lies_Status()
        Loop while (M1.Status OR M2.Status OR M3.Status = "running") Polling -> 2 Punkte
```

```
ERP.ID = “2”                    ‘----------Schritt 2
ERP.Teil = “Gehäuse”
ERP.Lies_teil()
M1.Variante = ERP.Variante                    M1 lesen -> 1 Punkt

ERP.ID = “1”
ERP.Teil = “Werk”
ERP.Lies_teil()
M2.Variante = ERP.Variante                    M2 lesen -> 1 Punkt

M3.Variante = 0
M1.Start_Modul()
M2.Start_Modul()
M3.Start_Modul()
Do                                            wieder Polling -> 1 Punkt
      M1.Lies_Status()
      M2.Lies_Status()
      M3.Lies_Status()
Loop while (M1.Status OR M2.Status OR M3.Status = “running”)


ERP.ID = “3”                    ‘----------Schritt 3
ERP.Teil = “Gehäuse”
ERP.Lies_teil()
M1.Variante = ERP.Variante

ERP.ID = “2”
ERP.Teil = “Werk”
ERP.Lies_teil()
M2.Variante = ERP.Variante

ERP.ID = “1”
ERP.Teil = “Deckel”
ERP.Lies_teil()
M3.Variante = ERP.Variante

M1.Start_Modul()
M2.Start_Modul()
M3.Start_Modul()
Do
      M1.Lies_Status()
      M2.Lies_Status()
      M3.Lies_Status()
Loop while (M1.Status OR M2.Status OR M3.Status = “running”)
```

Summe : 16 Punkte